20997

NATIONAL LIBRARY
OTTAWA

CANADA

BIBLIOTHÈQUE NATIONALE
OTTAWA

NAME OF AUTHOR.... JONATHAN CHARLES DESCHENEAU

TITLE OF THESIS.. MATHEMATICAL PROGRAMMING FOR PARAMETER OPTIMIZATION

UNIVERSITY... UNIVERSITY OF ALBERTA

DEGREE FOR WHICH THESIS WAS PRESENTED... MASTER OF SCIENCE

YEAR THIS DEGREE GRANTED... 1974

Permission is hereby granted to THE NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(Signed)... J. Descheneau

PERMANENT ADDRESS:

11747-83 Ave
Edmonton, Alberta
T6G 0V2

DATED.. 08 May 1974 ..19

NL-91 (10-68)

THE UNIVERSITY OF ALBERTA

MATHEMATICAL PROGRAMMING FOR PARAMETER OPTIMIZATION

by

© JONATHAN CHARLES DESCHENEAU

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

SPRING, 1974

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled "Mathematical Programming for Parameter Optimization" submitted by Jonathan Charles Descheneau in partial fulfillment of the requirements for the degree of Master of Science.

.....
Supervisor

.....

.....

.....

Date: 18 September, 1973

ABSTRACT

Nonlinear mathematical programming techniques and their application to the solution of selected optimal control problems are reviewed, implemented and evaluated.

Pertinent techniques for unconstrained minimization problems are discussed and applied to a set of test problems. The same procedure is followed for a set of constrained problems. Performance data is given for all test problems.

Two means of transforming the optimal control problem into a mathematical programming problem are outlined. The mathematical programming techniques are applied to the solution of two optimal control problems, the Earth-Mars rocket problem and the minimum energy regulator subject to linear constraints. The relative performances of the techniques are then evaluated.

ACKNOWLEDGEMENTS

The author wishes to acknowledge gratefully the supervision of Dr. J.F. Hauer. Drs. T.Y. Cheung, J. Tartar and D.E. Seborg are thanked for acting on the thesis committee.

Financial support was generously given by the Department of Computing Science and the National Research Council.

The author greatly appreciates the patience and support of his wife Catherine.

TABLE OF CONTENTS

CHAPTER			PAGE
CHAPTER I	INTRODUCTION		1
1.1	Historical Background		1
1.2	Scope of the Thesis		3
CHAPTER II	NUMERICAL OPTIMIZATION STRATEGIES		5
2.1	General Feasible Direction Method		5
2.2	Steepest Descents		7
2.3	Parallel Tangents		9
2.4	Fletcher-Reeves Conjugate Gradient		13
2.5	Generalized Newton-Raphson		14
2.6	Variable Metric Methods		16
	2.6.1	Davidon-Fletcher-Powell	16
	2.6.2	Self-scaling Variable Metric	16
	2.6.3	Revised Variable Metric	17
	2.6.4	Properties of the Variable Metric Methods	18
2.7	Optimization Along a Line		18
	2.7.1	Fibonacci and Golden Searches	19
	2.7.2	Cubic Interpolation	21
2.8	Case Studies		24
	2.8.1	Experimental Conditions	24
	2.8.2	Experimental Problems	26
	2.8.3	Experimental Results	28
2.9	Discussion and Conclusions		34
	2.9.1	General Remarks	34

2.9.2 Comparison of One Dimensional Search Techniques 35
2.9.3 The Effect of Initial Steepest Descents Steps on Convergence of the Algorithms 36
CHAPTER III CONSTRAINED OPTIMIZATION STRATEGIES 38
3.1 The Mathematical Programming Problem 38
3.2 The Kuhn-Tucker Conditions 39
3.3 The Constrained Problem Transformed to an Unconstrained Problem 41
3.4 Penalty-Barrier Methods 41
3.4.1 Penalty Constrained Methods 42
3.4.2 Barrier Constrained Methods 46
3.4.3 Combined Penalty-Barrier Problems 47
3.5 The Sequential Unconstrained Minimization Technique 47
3.6 Rosen's Gradient Projection Algorithm 49
3.6.1 The Projection Matrix 50
3.6.2 Rosen's Gradient Projection 52
3.6.3 Calculation Requirements of Gradient Projection 55
3.6.4 Calculation of the Search Direction d 59
3.7 Experimental Results 61
3.8 Experimental Problems 62
3.9 Numerical Studies 65

3.9.1 The Penalty Method 5
3.9.2 The Barrier Method 8
3.9.3 The SUMT Algorithm 71
3.9.4 Rosen's Gradient Projection Method 77
3.9.5 Conclusions 79
CHAPTER IV ORGANIZING THE OPTIMAL CONTROL PROBLEM FOR MATHEMATICAL PROGRAMMING SOLUTION 80
4.1 Formulation of the Optimal Control Problem 80
4.2 Solution of the Optimal Control Problem 81
4.2.1 Calculus of Variations 82
4.2.2 Dynamic Programming 83
4.3 Discrete Formulation of the Continuous Optimal Control Problem 84
4.4 Balakrishnan's Epsilon Technique 87
CHAPTER V CASE STUDIES 90
5.1 The Minimum Energy Regulator 90
5.1.1 Solution Using Rosen's Gradient Projection Method 93
5.1.2 Solution Using The SUMT Technique ... 94
5.2 Numerical Results 97
5.3 Earth-Mars Orbit Transfer Control Problem 99
5.3.1 Solution by the Modified Newton Raphson Method (MNR) 103
5.3.2 Solution by the MNR Method and

Golden Search 104

5.3.3 Solution by Other Mathematical Programming Techniques 104

5.3.4 A Variation to the Solution by the Modified Newton-Raphson Method 104

5.4 Numerical Results 105

CHAPTER VI SUMMARY AND CONCLUSION 118

6.1 Summary of Results 118

6.2 Conclusions 120

Bibliography 122

LIST OF TABLES

Table	Description	Page
2.1	Cubic interpolation	30
2.2	Golden Search	31
2.3	Effect of initial SD iterations	32
2.4	Effect of initial SD iterations	33
3.1	Penalty solutions r=0.2	66
3.2	Penalty solutions r=0.02	66
3.3	Penalty solutions r=0.002	67
3.4	Barrier solutions $r=10^{-1}$	69
3.5	Barrier solutions $r=10^{-2}$	69
3.6	Barrier solutions $r=10^{-3}$	70
3.7	SUMT solutions: Cubic interpolation	73
3.8	SUMT (cubic): Effect of restarting	73
3.9	SUMT solutions: Golden Search	74
3.10	SUMT (Golden): Effect of restarting	74
3.11	Test problem 13	75
5.1	Numerical results: MER	97

LIST OF FIGURES

Figure		Page
2.1	Steepest descent search	8
2.2	Zig-zagging of SD along a narrow valley	10
2.3	PARTAN search	12
3.1	Penalty method	43
3.2	Barrier method	44
3.3	The projection $z=P_q x$	51
3.4	Gradient projection as a function of two variables	53
3.5	Dropping a hyperplane	58
5.1	MER optimal control	95
5.2	MER optimal trajectory	96
5.3	Geometry of Earth-Mars orbit transfer	101
5.4	State trajectory x_1	106
5.5	State trajectory x_2	107
5.6	State trajectory x_3	108
5.7	Optimal control angle	109
5.8	Cost versus iterations - NR minimization	110
5.9	Cost versus iterations - DFP minimization	111
5.10	Error versus iterations - NR minimization	112
5.11	Error versus iterations - DFP minimization	113

TABLE OF SYMBOLS

Symbol	Description
$\cong$	equals approximately
$\neq$	does not equal
$\forall$	for all
$\rightarrow$	approaches
lim	limit
iff	if and only if
ϵ	is an element of
int	interior (of)
max,min	maximum, minimum
max	maximum over, with respect to, k
A^T	transpose of (matrix) A
$\|x\|$	norm of (vector) x
$\langle x,y \rangle$	inner product of x and y
E^n	real Euclidean n-space
x	decision vector (static MP problem)
x*	optimal value of x
f	objective function (static MP problem)
X	set of feasible x
d	direction vector
g	inequality-constraint function
h	equality-constraint function
$\nabla f(x)$	gradient of f w.r.t. X
λ	Lagrange multiplier vector
λ^*	value of λ satisfying K-T conditions for optimal x

$L(x,\lambda)$	Lagrangian function
r_k	(a) kth component of (vector) r
	(b) kth column vector of matrix R

CHAPTER I

1. Introduction

In the last two decades there has been considerable research and progress in the fields of optimal control and mathematical programming. Many techniques based upon the Calculus of Variations [17], Dynamic Programming [4], Pontryagin's Maximum Principle [32], and their extensions have been developed to cope with the numerical solution problem. But it is only in the last several years that direct mathematical programming techniques have been applied to the numerical solution of optimal control problems.

The aim of this thesis is to review, implement and evaluate standard nonlinear mathematical programming techniques and then compare their relative performance in the solution of certain mathematical programming and optimal control problems.

1.1 Historical Background

The mathematical programming (MP) problem involves the extremization of an objective function subject to inequality or equality constraints. The classical technique used to solve the equality constrained optimization problem is Lagrange's Method of Undetermined Multipliers. Even though some success has been achieved in adapting this method to problems involving inequality constraints, the trend has been to evolve computational methods of attack specialized

to unconstrained problems, notably the various gradient techniques.

In 1941 Courant [6] proposed another method of handling the constraints of functional minimization problems, now known as the method of penalty functions. This work was basically ignored until the early 60's when it received renewed interest. At this time gradient-type computational methods were developed, and, more recently, improved methods such as conjugate-gradient and second-variational methods have come into being. "Exact" methods of handling constraints including such techniques as Rosen's Gradient Projection [33] and Bryson and Denham's State Space Constraint Technique [8] have also been developed. The extension of penalty methods includes Fiacco and McCormick's Sequential Unconstrained Minimization Technique (SUMT) [9,10,11].

In the design of optimal control systems the classical method is the Calculus of Variations, an alternate view of which is Pontryagin's Maximum Principle. Another method which has found wide conceptual application is Bellman's Dynamic Programming [4]. Lately, however, the techniques of Mathematical Programming have been applied to the solution of optimal control problems, of particular value being the paper by Lasdon, Warren and Rice [26] which extended the application of nonlinear programming methods to a wider class of nonlinear control systems as compared to previous

authors. Recently there have been several papers applying Fletcher-Reeves conjugate-gradient method [14] and Davidon-Fletcher-Powell's variable metric algorithm [7,13] to control problems [25,27,31,36,43]. In addition Balakrishnan has developed another technique called the Epsilon Method [2,3], which lends itself very readily to solution by mathematical programming techniques as will be shown in this work.

1.2 Scope of the Thesis

Chapter II reviews the more pertinent techniques for solving unconstrained minimization problems, restricting itself to those techniques employing first order gradient information but with a brief mention of a second order technique. The algorithms for these methods are given and experimental results are presented at the end of the chapter.

Chapter III reviews the better known techniques for the optimization of constrained minimization problems with emphasis on the SUMT and Gradient Projection (GP) algorithms. The algorithms for these two methods are given with experimental results.

Chapter IV reviews some of the methods for solving the optimal control problem. It discusses briefly the methods of Calculus of Variations, the Minimum Principle, and Dynamic Programming. The discrete formulation of the

continuous optimal control problem and its solution by means of mathematical programming is discussed. Finally there is a discussion of Balakrishnan's Epsilon Technique and its solution using a Rayliegh-Ritz expansion.

Chapter V applies the methods implemented in Chapters II and III to the solution of certain control problems by means of the techniques discussed in Chapter IV.

In Chapter VI there is a summary of the results of previous chapters. Conclusions are drawn and recommendations for further study are proposed.

CHAPTER II

2. Numerical Optimization Strategies

In this chapter the methods of solving unconstrained minimization problems are reviewed. Emphasis is placed on descent methods which make use of first-order gradient information. Direct search techniques such as Rosenbrock's method are ignored.

The unconstrained minimization problem consists of finding the vector $x^*=(x_1^*,x_2^*,\ldots,x_n^*)$ which minimizes a scalar function $f(x)$. This unconstrained minimization problem is a special case of the Mathematical Programming problem (see Chapter III). The special feature of it is that the solution need not satisfy any constraints or conditions.

Section 2.7 of the chapter deals with techniques for minimization along a line. These are methods for solving optimization problems in one variable. Such methods are necessary for all of the search techniques dealt with in this chapter.

2.1 General Feasible Direction Method

Zoutendijk [45] gave the name "methods of feasible directions" to descent methods that generate a sequence of feasible directions. This category of methods, as defined, includes most of the search procedures in common use.

The general feasible direction method is defined as follows:

For small α, by a first order Taylor series expansion, given a direction of search d:

$$f(x+\alpha d)=f(x)+\alpha\langle\nabla f(x),d\rangle$$

And for a direction of descent d

$$f(x+\alpha d)<f(x)$$

is satisfied; d is "usable" at x if

$$\langle\nabla f(x),d\rangle < 0$$

Additionally, for constrained problems α must be sufficiently small that no constraints are violated.

Definition of Algorithm

(1) Select initial trial point $x^k=x^0 \in X$;

(2) Generate a feasible direction d^k corresponding to x^k;

(3) Find the value of $\alpha_k >0$ minimizing $f(x^k+\alpha_k d^k)$ and obtain the next trial point as

$x^{k+1}=x^k+\alpha_k d^k$;

(4) Set $k=k+1$ and repeat from step (2) as required.

Cost Factors:

This procedure solves the original vector minimization problem by generating a sequence of optimizations whose efficiency is dependent upon the algorithms used to generate α and d. In determining the cost of a particular algorithm one must consider the following:

(1) ease of programming

(2) storage requirements

(3) per-iteration costs

(4) rate of convergence

(5) faults or limitations

(6) advantages over other methods

The following sections describe the methods by which the parameters α and d are chosen.

2.2 Steepest Descent

From the Taylor series expansion of f(x) it can be seen that not only is $-\nabla f(x)$ (if $\nabla f \neq 0$) a usable direction but it is the locally steepest direction of descent. This first order gradient method, though it selects the locally steepest direction, possesses no other optimality properties such as finding the minimum in the least number of steps.

Definition of Algorithm

(1) Select initial trial point $x^k = x^0 \in X$;

(2) Generate a direction of search $d^k = -\nabla f(x^k)$

(3) Find the value of $\alpha_k > 0$ minimizing $f(x^k + \alpha_k d^k)$ and obtain the next trial point as

$x^{k+1} = x^k + \alpha_k d^k$;

(4) Set $k = k+1$ and repeat from step (2) as required.

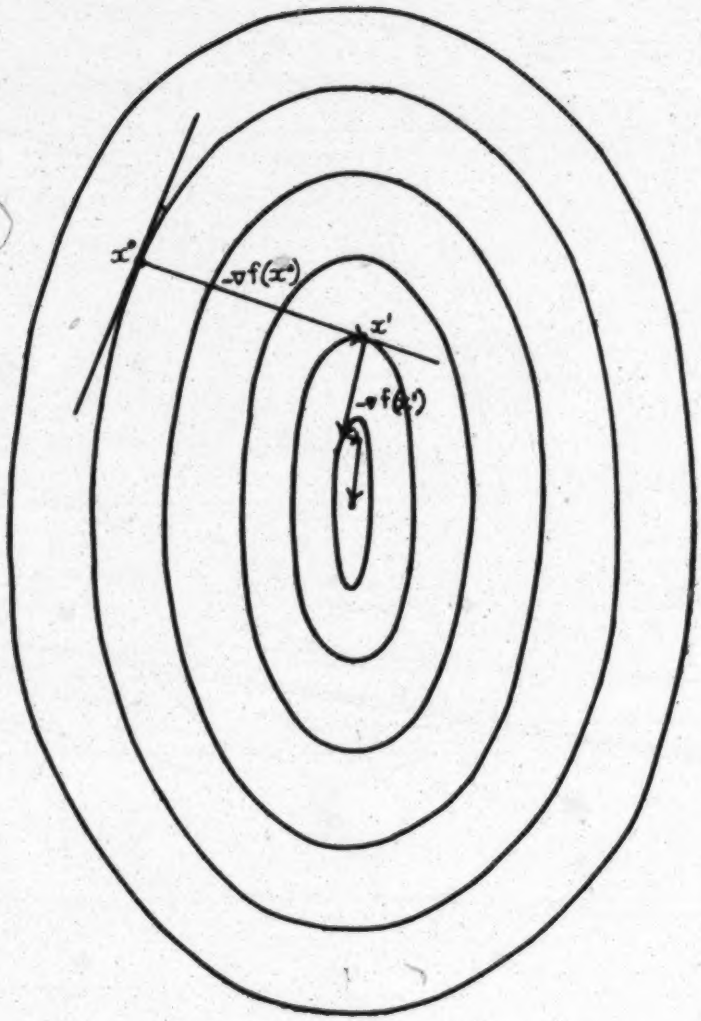

Figure 2.1 Steepest descent search

Cost Factors:

(1) Easily programmed and low storage requirement (of order n);

(2) Low per-iteration cost;

(3) Usually possesses a high initial rate of convergence;

(4) Prone to zig-zagging (see Figure 2.2), decelerating the rate of convergence as x^* is approached for functions which are more than slightly eccentric, thus requiring a large number of iterations.

2.3 Parallel Tangents (PARTAN) Procedure [35]

PARTAN is a technique of accelerating the steepest descents method.

If $f(x)$ is quadratic and $x \in E^2$, then the minimum $f(x^*)$ can be located in one step, after x^1 and x^2 are generated from x^0 by the steepest descent method. This is done by searching for the minimum along the ray defined by x^2-x^0. This gives rise to the following algorithm:

Definition of Algorithm

(1) Select initial trial point $x^k=x^{-1}=x^0 \in X$;

(2) Generate the direction of search d according to the following rules:

$$d^0=-\nabla f(x^0);$$

$$d^k=-\nabla f(x^k) \text{ for } k=1,3,5,7,\ldots;$$

$$d^k=x^k-x^{k-3} \text{ for } k=2,4,6,\ldots;$$

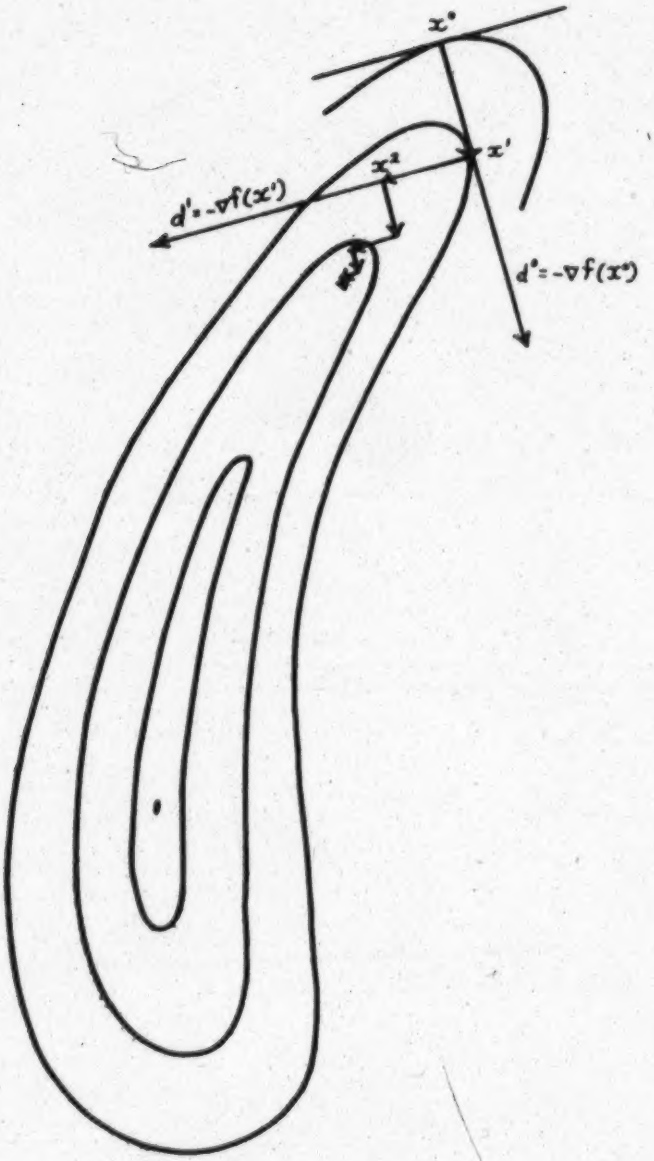

Figure 2.2 Zig-zagging of SD along a narrow valley

(3) Find the value of $\alpha_k > 0$ minimizing $f(x^k + \alpha_k d^k)$; and obtain the next trial point as $x^{k+1} = x^k + \alpha_k d^k$;

(4) Set $k = k+1$ and:

(a) for Continued Partan repeat from step(2) as required.

(b) for Iterated Partan continue from step(2) until $k = 2n+1$. Then re-initialize setting $x^k = x^{2n-1}$, $k=0$ and continue from step(2) as required.

Properties:

If $f(x)$ is a positive-definite quadratic

$$f(x) = x^T Q x + a^T x + b$$

then:[22]

(a) the gradient directions (d^0, d^1), (d^1, d^3), (d^3, d^5),... are orthogonal;

(b) the pair of vectors x^k, x^{k-2} $k=1,3,5,7,...$ are Q-conjugate;

(c) convergence occurs within 2n-1 iterations.

Cost Factors:

(1) low storage requirements (order n);

(2) low per iteration cost;

(3) possesses a good initial rate of convergence which tends to deteriorate as the minimum is approached for nonquadratics;

(4) requires a large number of iterations for eccentric functions.

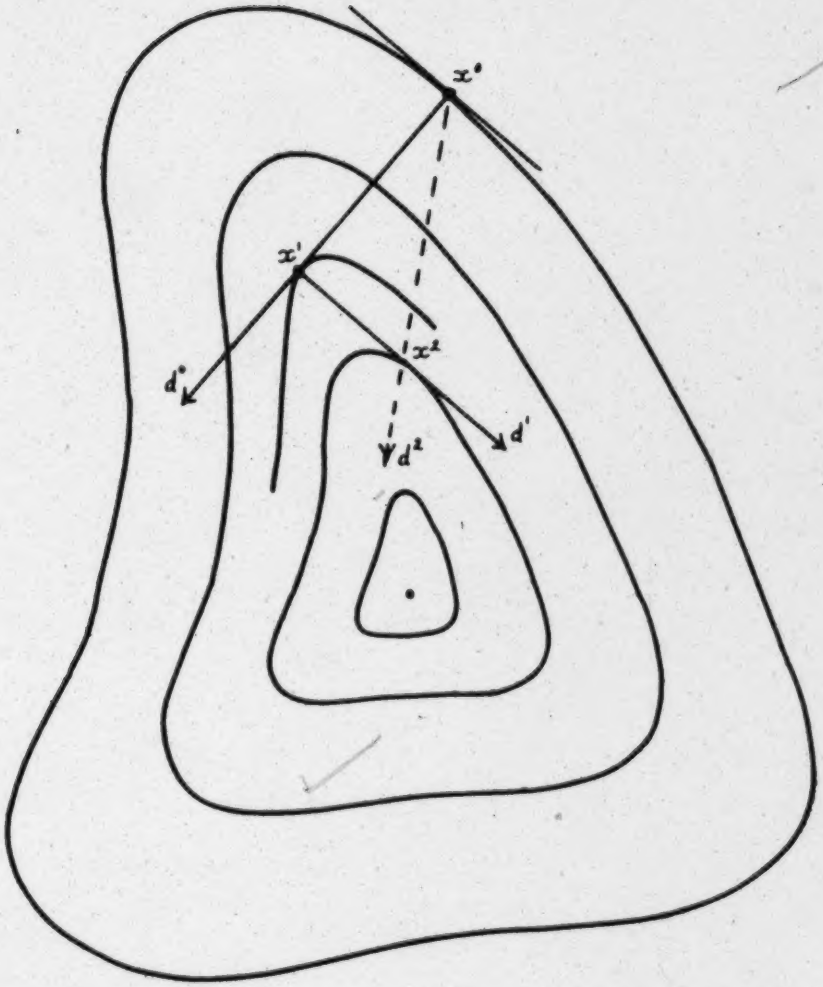

Figure 2.3 PARTAN search

2.4 Fletcher-Reeves Conjugate Gradient Alorithm [14]

This first-order gradient method uses a general form of the Gram-Schmidt orthogonalization process to generate a set of conjugate descent directions. The method converts $-\nabla f(x^k)$ into the direction d^k such that it becomes Q-conjugate with the previously generated d's. These directions are generated by:

$$d^0 = -\nabla f(x^0)$$

$$d^k = -\nabla f(x^k) + \sum_{i=1}^{k} \beta^i d^{i-1} \qquad k>1$$

the β^i's are chosen to satisfy the condition that d^k should be conjugate to all previous directions. The general formula for β^i can be established [22] as

$$\beta^i = \langle \nabla f(x^k), \nabla f(x^k) \rangle / \langle \nabla f(x^{k-1}), \nabla f(x^{k-1}) \rangle \qquad i=k$$

$$\beta^i = 0 \qquad i \neq k$$

Practical experience has shown that the performance of the method is improved if it is restarted from time to time due to contamination of $\beta\, d^{k-1}$ by computational error. However, the restarting scheme often turns out to be problem dependent. Still, restarting every n+1 iterations is generally the best scheme [14,15,22].

Properties:

If $f(x)$ is a positive definite quadratic

$$f(x) = x^T Q x + a^T x + b$$

then:

(a) the generated directions are Q-conjugate;

(b) convergence occurs within n iterations;

However, due to round-off in the arithmetic process and inaccuracies in determining α, it usually requires more than n iterations and the procedure must be restarted according to some formula.

Definition of Algorithm

(1) Select initial trial point $x^k = x^0 \in X$;

(2) Generate a direction of search according to:

$$d^0 = -\nabla f(x^0)$$

$$d^k = -\nabla f(x^k) + \beta d^{k-1} \quad k = 1, 2, 3, \ldots$$

Where $\beta = \langle \nabla f(x^k), \nabla f(x^k) \rangle / \langle \nabla f(x^{k-1}), \nabla f(x^{k-1}) \rangle$;

(3) Find the value of $\alpha_k > 0$ minimizing $f(x^k + \alpha_k d^k)$ and obtain the next trial point as

$x^{k+1} = x^k + \alpha_k d^k$;

(4) Set $k = k+1$ and repeat from step (2) as required. If $k = n+1$ then reinitialize setting $k = 0$, and $x^0 = x^{k+1}$ and continue from step (2).

Cost Factors

(1) low storage requirements (order n);

(2) low per iteration costs;

(3) requires very accurate determination of α;

(4) sensitive to re-start policy, which is usually problem dependent and for which no set guideline exists.

2.5 Generalized Newton-Raphson Method [12]

This second-order method makes use of the Hessian, A(x), of f(x) in computing the directions of search. A(x)

must be positive-definite.

Consider a second order expansion of $f(x)$ about x^k:

$$f(x) \tilde{=} f(x^k) + (x-x^k)^T \nabla f(x^k) + \tfrac{1}{2}(x-x^k)^T A(x^k)(x-x^k)$$

If this is differentiated with respect to x, one obtains

$$\nabla f(x) = \nabla f(x^k) + A(x^k)(x-x^k)$$

If this is computed at x^{k+1}, then, trying to maximize $f(x^k) - f(x^{k+1})$ at the k^{th} iteration, one picks x^{k+1} so that $f(x^{k+1})=0$. This implies

$$x^{k+1} = x^k - A^{-1}(x^k)\nabla f(x^k)$$

If $f(x)$ is a quadratic, then a single step of unit length along the direction gives the solution.

The method is very effective for a large class of problems. For difficulties and improvements of the method see [12].

Definition of Algorithm

(1) Select initial trial point $x^k = x^0 \in X$;

(2) Generate a direction of search

$$d^k = -A^{-1}(x^k)\nabla f(x^k);$$

(3) Find the value of $\alpha_k > 0$ minimizing $f(x^k + \alpha_k d^k)$ and obtain the next trial point as

$$x^{k+1} = x^k + \alpha_k d^k;$$

(4) Set k=k+1 and repeat from step (2) as required.

Cost Factors:

(1) converges to a minimum very rapidly for a large class of problems;

(2) A^{-1} may not always exist or may be ill-conditioned;

(3) in a non-convex problem f(x) may not decrease when x is not near the minimum;

(4) it is often tedious or impossible to calculate the analytic forms for A;

(5) high storage requirements of order $(n^2/2)$;

(6) high per-iteration cost due to matrix inversion required for $A^{-1}(x)$.

2.6 Variable Metric or Quasi-Newton Methods

These methods make use of first order gradient information, assume that the function is at least locally quadratic, and usually attempt to estimate A^{-1}. The methods consist of calculating an n n matrix H and forming a search direction d by:

$$d^k = -H^k \nabla f(x^k)$$

(for later convenience define $y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$).

2.6.1 Davidon-Fletcher-Powell (DFP) [7,13]

In this version H^0 is any symmetric positive-definite matrix, usually the identity matrix I. The iterative formula for calculating H^{k+1} is:

$$H^{k+1} = H^k + \alpha \left[\frac{dd^T}{y^T d}\right]^k - \left[\frac{(Hy)(Hy)^T}{y^T H y}\right]^k$$

2.6.2 Self-scaling Variable Metric (SSVM) [29]

In this version H^0 is any symmetric positive-definite matrix, usually the identity matrix I. The iterative

formula for calculating H^{K+1} is:

$$H^{K+1} = \alpha \left[\frac{y^T d}{y^T H y}\right]^K \left[H - \frac{(Hy)(Hy)^T}{y^T H y}\right]^K + \alpha \left[\frac{dd^T}{y^T d}\right]^K$$

2.6.3 Revised Variable Metric (RVM) [12]

In this version H^0 is any positive semi-definite matrix, usually the identity matrix I. The iterative formula for H^{K+1} is:

$$H^{K+1} = H^K + \left[\frac{(\alpha d - Hy)(\alpha d - Hy)}{(\alpha d - Hy)^T y}\right]^K$$

2.6.4 Properties of the Variable Metric Methods [12,13,29]

For a quadratic these methods reach the minimum point x^* and, in general, H converges to A^{-1} in n iterations.

SSVM has the following properties in addition to those of DFP: Invariance to scaling of the function or variables, apparently better convergence for large n(>6), and less sensitivity to round-off errors and to the accuracy of the one-dimensional search for α. H does not converge to A^{-1}.

Fiacco and McCormick [12] state the following for the RVM method: Because H is positive semi-definite, one is allowed to use the variable metric method on large problems with a structure that makes it convenient or possible to compute second partial derivatives for some of the variables and not for others. This may be done in the following fashion:

$$A=\nabla^2 f=\begin{bmatrix} A & a^T \\ a & b \end{bmatrix}$$

$$\text{Let } H^0=\begin{bmatrix} A^{-1} & 0 \\ 0 & 0 \end{bmatrix}$$

and the change from H^k to H^{k+1} is as stated in the updating formula for RVM method. This advantage allows H to approach A^{-1} in fewer than n steps. This method is not as sensitive to the accuracy of the search for α as DFP.

For other schemes of updating the appoximation H see [21,37,38].

Cost Factors:

(1) Fast second-order convergence near a minimum and finite convergence for a quadratic functional;

(2) storage of order $(n^2/2)$;

(3) faster convergence for general non-quadratic functions when compared with other quadratically convergent methods;

(4) first-order method so there is no need to calculate second-order derivatives;

(5) the methods are basically stable;

(6) the methods may become unstable if the Hessian is singular near a solution, but this can usually be avoided if the methods are restarted;

(7) DFP and most of the variable metric methods require very accurate determination of α.

2.7 Optimization Along a Line

The one common feature that all the above methods have is that they require a search for local minima along each in a sequence of directions. Since the methods call for numerous linear searches it is very important to do them simply and efficiently.

The one-variable optimization problem in E^1 is: Given x^k and a search direction d^k find α_k such that

$$f(x^k+\alpha_k d^k)=\min_{\alpha} f(x^k+\alpha d^k)$$

In general, the methods search for α_k in some bounded interval [L,U] using some strategy to approximate α_k or a subinterval containing α_k. The upper and lower limits, L and U, can be found by a number of different methods: Quadratic extrapolation and/or interval doubling, using an increasing power of the Fibonacci ratio, or simple multiples of the previous upper limit.

The efficiency of the linear search is very dependent on the efficiency of the method used to determine U, which in turn is very problem-dependent. All of the above methods can be modified slightly to increase their efficiency in specific problems.

2.7.1 Fibonacci and Golden Searches [12,22]

Of the sequential search techniques, the theoretically most efficient algorithm for finding the subinterval

containing the minimum of a unimodal function is called the Fibonacci search and its simplified version is the Golden Section method. It is optimal in that it gives the smallest ratio of final to initial interval for a fixed number of function evaluations.

The Golden Section method is a numerical approximation to the Fibonacci search and is not optimal in the same sense. It is preferred because the number of function evaluations need not be specified in advance and the Fibonacci numbers need not be computed or stored in the program.

Definition of Algorithm

(1) Set the lower bound L=0
and the initial upper bound U=0.
Set counter k=0;

(2) Find an upper bound on U^k using the limiting Fibonacci ratio $1.618 \cong (1+\sqrt{5})/2$ [12].
Let $U^k=U^{k-1}+(1.618)^k$;

(3) If $f(x+U^k d)>f(x+U^{k-1}d)$ then U^k is the upper bound and go to step(4);
else return to step (2);

(4) Calculate $T_a=0.382(U-L)+L$ and $f_a=f(x+T_a d)$
$T_b=0.618(U-L)+L$ and $f_b=f(x+T_b d)$
go to step(7);

(5) Calculate $T_a=0.382(U-L)+L$ and $f_a=f(x+T_a d)$
go to step (7);

(6) Calculate $T_b=C.618(U-L)+L$ and $f_b=f(x+T_b d)$
go to step (7);

(7) If $f_a>f_b$ set $L=T_a$, $T_a=T_b$, $f_a=f_b$,
and go to step (6).
Or else set $U=T_b$, $T_b=T_a$, $f_b=f_a$,
and go to step (5).

If $\|T_b-T_a\| < \epsilon$ then the interval has been reduced to the required accuracy and the algorithm is terminated with $\alpha=(T_b+T_a)/2$

Other termination criteria can also be used. [22].

Cost Factors:

The Golden Search is second only to the Fibonacci Search, from which it is derived, as a sequential univariate search procedure that guarantees a prescribed accuracy. However, if ϵ is small, a large number of function evaluations are required. Because of this, methods based on curve fitting have been found less costly and to give better accuracy. The Fibonacci and Golden Searches are recommended mainly when the objective function is irregular and approximation by a low-order polynomial is likely to be inadequate (such as sometimes occurs in SUMT [12]).

2.7.2 Cubic Interpolation [7,13,15]

The minimum along the line can also be approximated by curve fitting. This consists of representing the function

along a line by a simple pcynomial and using its minimum to approximate that of the function. This allows us to replace a complex function by a simple function whose minimum is known analytically. A very precise fit is not needed, therefore the function can be approximated by a low-order (second or third) polynomial. This assumes that the function is smooth and unimodal in the interval being considered. In most practical applications the selection of a smooth enough interval assures us of the validity of this assumption.

Where function gradients are available Davidon's method of cubic interpolation is the method chosen to locate the minimum along a descent direction. This method requires the value of the function and its gradient at two points which bracket the minimum.

Definition of the Algorithm :

(1) $R1=\langle \nabla f, d \rangle$

(2) Quadratic extrapolation is used to find an initial estimate of the upper limit of α.

$$\eta = -2(f'-f)/R1$$

where f' is the value of f from the previous DFP iteration.

Another approximation to the upper limit of α is

$$R0=1/(\langle d,d \rangle)$$

Then let $\alpha=\min(R0, \eta, 16)$

$\alpha'=0$, $f'=f$

(3) $y = x + \alpha d$

(4) If $f(y) > f(x)$ or $\langle \nabla f(y), d \rangle = R2 > 0$

Then go to step (6)

(5) $\alpha' = \alpha$

$R1 = R2$

$\alpha = 2\alpha$

$f' = f(y)$

go to step (2)

(6) $h = \alpha - \alpha'$

$z = 3(f' - f(y))/h + R1 + R2$

$w = (\max(z^2 - R1 * R2, 0))^{1/2}$

$\alpha'' = h(1 - (R2 + w - z)/(R2 - R1 + 2w))$

$\alpha'' = \alpha'' + \alpha'$

(7) $x = x + \alpha'' d$

Calculate $f(x)$ and $\nabla f(x)$

(8) If $f(x) > \min(f', f(y))$ then the interpolation is repeated in order to obtain better accuracy. The subinterval chosen depends on the sign of $\langle \nabla f(x), d \rangle$

If negative the interval (α'', α) is used, if positive the interval (α', α'') is used.

The value of α'' thus obtained is accepted as a good approximation to the minimum.

Cost Factors:

Repeated interpolation using the cubic method is generally more efficient than the Golden method. The efficiency and accuracy is particularly good if the

objective function is smooth enough to be closely approximated by a cubic polynomial. The method can fail if the objective function is not smooth enough. If great accuracy is required, the computational cost of repeated approximation and calculation of gradients will decrease the effectiveness of this method.

If the gradients are very expensive to calculate or unavailable then the Golden method of interpolation could be used. Davies' algorithm, or Powell's method could alternatively be used as the method of minimization (see [1,22] for explanation of the latter two methods).

2.8 Case Studies

In this section the results of applying the unconstrained minimization techniques presented earlier in sections 2.2 to 2.7 to a set of unconstrained problems are presented. Most of the test problems are standard in the literature and can be found in references [12,13,21,22,30,35]. The computer code used in the solution of these problems and a guide to its use may be found in [49].

2.8.1 Experimental Conditions

In each case the function $f(x)$ is to be minimized, x^0 is the starting point and x^* is the optimum point which minimizes $f(x)$ over the feasible set X.

Starting Conditions: Each test problem has its own starting point x^0. For the variable metric algorithm the initial metric H^0 is taken to be the identity matrix I.

Stopping Conditions: An algorithm is stopped when all the following inequalities are satisfied:

$$d^T d \leq \epsilon$$

$$\alpha^2 d^T d \leq \epsilon$$

$$\alpha \| d_i \| \leq \epsilon \qquad i=1,2,\ldots,n$$

ϵ was taken to be 10^{-6}

One Dimensional Search: In the cubic interpolation algorithm it is sometimes desireable to check if the interpolation is accurate enough. This can be done by checking if the vector product

$$\langle d^i, \nabla f(x^{i+1}) \rangle \leq \epsilon'$$

If this inequality is not satisfied the interpolation is done again over a refined interval. However, for the problems under consideration this was found to be unnecessary and was omitted. The extra calculations and function evaluations required resulted in only a small decrease in the number of iterations and in a large increase in the total time.

The Golden search was stopped when

$$\| T_b - T_a \| < \epsilon$$

ϵ was taken to be 10^{-2} for both the unconstrained and constrained problems. A smaller value of epsilon resulted in fewer iterations but more function evaluations and

generally an increase in the total time.

2.8.2 Experimental Problems

The problems used to test the programs implementing the algorithms described above are given below. Each problem is given a number and is referred to by that number.

TP1: (2-variable quadratic)

$f(x) = x_1^2 + 5x_2^2$

$x^0 = (2,3)$

$x^* = (0,0)$

$f(x^*) = 0.0$

TP2: (3-variable quadratic)

$f(x) = x_1^2 + 2x_2^2 + 3x_3^2$

$x^0 = (1,1,1)$

$x^* = (0,0,0)$

$f(x^*) = 0.0$

TP3: (2-variable hyperellipse)

$f(x) = x_1^4 + x_2^2 - 3x_1 - 2x_2 + 2$

$x^0 = (0,0)$

$x^* = (0.90856, 1.0)$

$f(x^*) = -1.04426$

TP4: (Fiacco and McCormick [11])

$f(x) = (x_1^4 - 3)^2 + x_2^4$

$x^0 = (10^{-3}, 10^2)$

$x^* = (3^{\frac{1}{4}}, 0.0) \cong (1.316074, 0.0)$

$f(x^*) = 0.0$

The function has a stationary point $f(x) = 9.0$ at $x = (0,0)$

TP5: (Beale's Function [28])

$$f(x) \triangleq (1.5-x_1(1-x_2))^2+(2.25-x_1(1-x_2^2))^2 + (2.625-x_1(1-x_2^3))^2$$

$x^0=(1,0.8)$

$x^*=(3,0.5)$

$f(x^*)=0.0$

The graph of this function exhibits a narrow valley, with its bottom a curve tending towards the line $x_2=1$.

TP6: (Rosenbrock's Function [12])

$$f(x)=100(x_1^2-x_2)^2+(1-x_1)^2$$

$x^0=(-1.2,1)$

$x^*=(1,1)$

$f(x^*)=0.0$

The graph of $f(x)$ is distinguished by a banana-shaped valley, its bottom curve given by $x_1^2=x_2$.

TP7: (Fletcher-Powell [12])

$$f(x)=100((x_3-10\omega)^2+(r-1)^2)+x_3^3$$

where

$x_1=r\cos(2\pi\omega)$

$x_2=r\sin(2\pi\omega)$

$r=(x_1^2+x_2^2)^{1/2}$

$x^0=(-1,0,0)$

$x^*=(1,0,0)$

$f(x^*)=0.0$

This function defines a helical valley, with axis in the x_3 direction.

TP8: (Powell's Function [12])

$$f(x) = (x_1+10x_2)^2+5(x_3-x_4)^2+(x_2-2x_3)^4+(10x_1-x_4)^4$$

$$x^0 = (3,-1,0,1)$$

$$x^* = (0,0,0,0)$$

$$f(x^*) = 0.0$$

This function is distinguished by a singularity of $f_{xx}(x^*)$; it is not positive definite. It was designed especially for testing algorithms with a quadratically converging search.

TP9: (Wood's Function [11])

$$f(x) = 100(x_1^2-x_2)^2+(1-x_1)^2+90(x_3^2-x_4)^2$$
$$+(1-x_3)^2+10.1((1-x_2)^2+(1-x_4)^2)$$
$$+19.8(1-x_2)(1-x_4)$$

$$x^0 = (-3,-1,-3,-1)$$

$$x^* = (1,1,1,1)$$

$$f(x^*) = 0.0$$

This problem has a stationary value of $f(x)=7.876$ at $x=(-0.9679,\ 0.9471,-0.9695,\ 0.9512)$

2.8.3 Experimental Results

The results of applying the minimization algorithms to the above test problems are given in the tables below. These results allow conclusions to be drawn concerning the relative effectiveness of the algorithms.

Each algorithm is applied to each test problem and the following three parameters are recorded:

(1) IT- the number of iterations it took for the method to converge.

(2) #F- the total number of function evaluations taken by the algcrithm.

(3) Time- the time in seconds that it took for an IBM 360/67 to solve the problem.

The names of these algcrithms are abbreviated as follows:

SD- Steepest Descent

GNR- Generalized Newton-Paphson

IPTAN- Iterated PARTAN

CPTAN- Ccntinued PARTAN

CG- Conjugate Gradients

DFP- Davidon-Fletcher-Powell algorithm

SSVM- Self-Scaling Variable Metric algorithm

RVM- Revised Variable Metric

TEST	PROB	SD	IPTAN	CPTAN	CG	DFP	SSVM	RVM
	IT	12	4	4	3	3	3	3
TP1	#F	28	12	12	10	10	10	10
	TIME	0.071	0.055	0.053	0.052	0.054	0.055	0.057
	IT	21	6	6	4	4	4	4
TP2	#F	44	15	15	10	10	10	10
	TIME	0.088	0.06	0.061	0.055	0.061	0.062	0.062
	IT	29	10	21	9	6	6	6
TP3	#F	60	22	48	20	14	14	14
	TIME	0.099	0.064	0.087	0.061	0.062	0.063	0.062
	IT	11	12	16	1.	8	33	38
TP4	#F	70	77	100	99	58	165	152
	TIME	0.082	0.099	0.113	0.113	0.089	0.187	0.179
	IT	564	25	23	15	9	11	10
TP5	#F	1132	64	65	48	24	32	24
	TIME	1.206	0.109	0.106	0.089	0.076	0.085	0.077
	IT	>4000	25	318	27	27	24	24
TP6	#F	>8011	68	760	66	70	86	59
	TIME	>7.492	0.102	0.734	0.104	0.126	0.131	0.114
	IT	1306	61	43	28	23	23	22
TP7	#F	2616	177	121	78	58	66	58
	TIME	3.891	0.290	0.214	0.154	0.161	0.168	0.155
	IT	4000	114	148	50	45	33	10
TP8	#F	8002	372	420	139	263	115	26
	TIME	10.181	0.457	0.528	0.213	0.391	0.245	0.100
	IT	3750	30	67	36	44	26	42
TP9	#F	7506	91	1.7	111	1.0	89	115
	TIME	10.131	0.160	0.291	0.187	0.350	0.211	0.278

Table 2.1 Cubic interpolation

TEST	PROB	SD	IPTAN	CPTAN	CG	DFP	SSVM	RVM
	IT	27	15	21	9	5	6	5
TP1	#F	352	1.6	286	118	78	96	78
	TIME	0.148	0.106	0.128	0.083	0.073	0.079	0.076
	IT	21	16	15	8	6	8	6
TP2	#F	274	209	220	105	88	123	88
	TIME	0.140	0.121	0.128	0.088	0.086	0.101	0.087
	IT	29	9	1.	9	6	6	6
TP3	#F	378	133	263	118	88	92	89
	TIME	0.164	0.091	0.129	0.085	0.079	0.083	0.083
	IT	*	*	*	*	5	5	5
TP4	#F					199	235	199
	TIME					0.109	0.123	0.110
	IT	>500	23	51	26	9	10	10
TP5	#F	>6529	339	703	357	146	164	158
	TIME	>2.418	0.175	0.314	0.184	0.110	0.117	0.115
	IT	>400	89	49	335	14	15	13
TP6	#F	>9999	2066	1091	7896	303	370	274
	TIME	>2.922	0.645	0.370	2.367	0.148	0.170	0.140
	IT	>350	210	79	67	21	21	1.
TP7	#F	>6720	4247	1534	1444	396	413	325
	TIME	>4.700	3.037	1.127	1.045	0.148	0.365	0.301
	IT	>350	1.4	956	168	23	46	31
TP8	#F	>4576	2772	>9999	2206	491	855	516
	TIME	>2.200	1.369	5.153	1.106	0.323	0.551	0.353
	IT	>350	541	480	209	34	23	>700
TP9	#F	>8950	>9999	>9999	5127	884	509	>9999
	TIME	>4.289	5.606	5.153	2.528	0.544	0.349	>16.15

* - METHOD DID NOT CONVERGE TO CORRECT SOLUTION

Table 2.2 Golden Search

TEST PROBLEM		IPTAN NORMAL	IPTAN 3SD	CPTAN NORMAL	CPTAN 3SD	CG NORMAL	CG 3SD
TP1	IT	4	7	4	7	3	3
	#F	12	18	12	18	10	10
	TIME	0.055	0.060	0.053	0.058	0.052	0.054
TP2	IT	6	9	6	9	4	4
	#F	15	20	15	20	10	10
	TIME	0.060	0.064	0.061	0.066	0.055	0.058
TP3	IT	10	10	21	16	9	8
	#F	22	22	48	34	20	18
	TIME	0.064	0.063	0.087	0.076	0.061	0.060
TP4	IT	12	17	16	20	19	20
	#F	77	85	100	94	99	110
	TIME	0.099	0.106	0.113	0.111	0.113	0.120
TP5	IT	25	25	23	89	15	14
	#F	64	67	65	214	48	44
	TIME	0.109	0.108	0.106	0.261	0.089	0.085
TP6	IT	25	29	318	35	27	29
	#F	68	75	760	91	66	77
	TIME	0.102	0.108	0.734	0.122	0.104	0.112
TP7	IT	61	64	43	81	28	27
	#F	177	187	121	243	78	83
	TIME	0.290	0.301	0.214	0.374	0.154	0.160
TP8	IT	114	68	148	119	50	68
	#F	372	213	420	322	139	1.4
	TIME	0.457	0.283	0.528	0.426	0.213	0.277
TP9	IT	30	83	67	153	36	37
	#F	91	270	1.7	370	111	144
	TIME	0.160	0.368	0.291	0.532	0.187	0.191

Table 2.3 Effect of initial SD iterations

TEST PROBLEM		DFP NORMAL	DFP 3SD	SSVM NORMAL	SSVM 3SD	MDFP NORMAL	MDFP 3SD
TP1	IT	3	6	3	6	3	6
	#F	10	16	10	16	10	16
	TIME	0.054	0.062	0.055	0.062	0.057	0.062
TP2	IT	4	7	4	7	4	7
	#F	10	16	10	16	10	16
	TIME	0.061	0.070	0.062	0.071	0.062	0.066
TP3	IT	6	8	6	8	6	8
	#F	14	18	14	18	14	18
	TIME	0.062	0.066	0.063	0.068	0.062	0.066
TP4	IT	8	14	33	33	38	12
	#F	58	88	165	163	152	79
	TIME	0.089	0.118	0.187	0.188	0.179	0.104
TP5	IT	9	12	11	12	10	12
	#F	24	30	32	29	24	30
	TIME	0.076	0.084	0.085	0.083	0.077	0.083
TP6	IT	27	21	24	31	24	25
	#F	70	60	86	101	59	63
	TIME	0.126	0.111	0.131	0.149	0.114	0.117
TP7	IT	23	20	23	23	22	20
	#F	58	49	66	58	58	50
	TIME	0.161	0.144	0.168	0.161	0.155	0.143
TP8	IT	45	25	33	52	10	25
	#F	263	94	115	175	26	73
	TIME	0.391	0.202	0.245	0.358	0.100	0.184
TP9	IT	44	25	26	25	42	25
	#F	1.0	68	89	85	115	66
	TIME	0.35	0.191	0.211	0.207	0.278	0.182

Table 2.4 Effect of initial SD iterations

2.9 Discussion and Conclusions

From the tables it can be seen that the results are basically problem-dependent, but that certain general conclusions can be drawn.

2.9.1 General Remarks

The three variable metric algorithms and the conjugate gradient algorithm solved the two quadratic test problems in n iterations. (The table of results shows n+1 because that is the minimum allowed by the program though the methods actually converged earlier.) Both of the forms of PARTAN required 2n iterations for these problems. These results agree with the theoretical expectations for these algorithms for quadratic functions. Steepest descent required many more iterations but because of its simpler logic which requires fewer calculations, the total time required for solution is only 40% higher.

It is in the solution of the non-quadratic problems that the greater eficiency of the accelerator techniques becomes apparent. From Tables 2.1 and 2.2 it can be seen that in general SD is less efficient in terms of number of iterations and total time by one order of magnitude. It is also apparent that PARTAN is much less efficient than the other techniques.

Conjugate-gradients and the three variable metric

algorithms are approximately equal in performance in terms of number of iterations and time. C-G generally takes more iterations but less time because of its simpler algorithm. The variable metrics are generally equal in performance but each has at least one problem in which it excels.

For test problems 4 to 9 the performance of all the algorithms was found to be very sensitive to the mode of implementation of the one-dimensional search algorithms. Their performance can be varied by factors of 2 or 3 both adversely and favourably. All numbers in the tables were obtained from the same program, that implementation being the one that gave the best results in general.

2.9.2 Comparison of One Dimensional Search Techniques

Two methods of one-dimensional search were used: Golden Search and Davidon's cubic interpolation.

From Table 2.2 it can be seen that for the same number of iterations the Golden method takes 25% to 100% longer with 4 to 6 times as many function evaluations as the cubic method. The only saving here is that only the final function evaluation of each iteration requires the calculation of the gradient. For several of the examples an ϵ of 10^{-2} is not accurate enough to achieve satisfactory results; but decreasing ϵ increases the number of function evaluations per iteration so that while there would be a decrease in the number of iterations there would not be a

corresponding decrease in time.

In general, one can say that the cubic interpolation algorithm gives more rapid and dependable convergence than the Golden Search. In addition, the performance of the Golden Search is very dependent upon the individual problem and the choice of ϵ for which no a priori guidelines exist. The main advantage of the Golden Search is that it does not require the calculation of gradients and its accuracy is not dependent upon the sometime questionable accuracy of a low-order polynomial fit.

2.9.3 The Effect of Initial Steepest Descents Steps on Convergence of the Algorithms

Since the SD method has a high initial rate of convergence, and since the convergence of the variable metric techniques is sometimes hampered by contamination of the metric due to poor initial search directions, it was decided that trying a few initial iterations of SD at the beginning of each search might improve the rate of convergence. Tables 2.3 and 2.4 present the results of trying three initial iterations of SD.

It was found through experiment that three iterations was the best number. Less made little difference and more often hampered convergence though sometimes convergence was improved.

From the tables one can see that SSVM and conjugate gradients are relatively unaffected; both methods of PARTAN and RVM are both adversely and beneficially affected. However, DFP generally benefits from the initial iterations of SD searches.

In conclusion, from the evidence one gains the impression that the effect of initial iterations of SD searches is problem dependent but that the DFP algorithm usually benefits from it.

CHAPTER III

3. Constrained Optimization Strategies

In this chapter some of the methods of solving constrained minimization problems are reviewed. Emphasis is placed on the SUMT algorithm of Fiacco and McCormick and on Rosen's Gradient Projection Algorithm.

In the real world one cannot always solve a minimization problem without running into constraints. In dynamic-system design, for example, there are usually limits on loads, velocities, accelerations, etc., which result in constraints being placed on the solutions. Unconstrained methods cannot cope directly with these limitations and so different techniques or reformulations of existing techniques must be used to solve these problems.

3.1 The Mathematical Programming Problem [20]

The MP problem involves the minimization of an objective function, this function being subject to inequalities or equalities called constraints. In general, the objective function and constraints may be nonlinear functions in all or some of the variables. The special case of no constraints was handled earlier in Chapter 2.

Any set of variables which satisfies all the constraints must be a feasible solution, and the feasible

solution which optimizes the objective function is the optimal solution to the NP problem.

The general form of the NP problem is:

"Minimize $f(x)$

subject to $g_i(x) \leq 0 \qquad i=1,\ldots,m$

$h_i(x) = 0 \qquad i=m+1,\ldots,q$"

The scalar objective function $f(x)$ is a function of the n-dimensional vector x.

There are several special classes of the NP problem. If $f(x)$, $g(x)$ and $h(x)$ are all linear then one has the linear programming (LP) problem. If any of $f(x)$, $g(x)$ or $h(x)$ are nonlinear then one has the non-linear programming (NLP) problem. The quadratic programming (QP) problem, a special case of the NLP problem, has a quadratic objective function with linear constraints. There are other special types such as the geometric programming problem, integer programming problem and stochastic programming problem: They are not considered here. There exist special techniques for the solution of the QP and LP problems but they are not involved in this study.

3.2 The Kuhn-Tucker Conditions [20,44]

Consider the NLP problem in the form stated above. If x^* is an optimal solution to the NLP problem, then the Kuhn-Tucker (K-T) conditions (under reasonable assumptions) are necessarily satisfied. Generally the K-T conditions state

that given the point x*, if one moves in any direction d a distance $\alpha \| d \|$ ($\alpha > 0$), as long as one remains in the feasible region the objective function will not decrease. In addition "The K-T theorem is intimately related to the Lagrange multiplier theorem, which it generalizes,... either theorem can be used to obtain the other."[20]

The K-T conditions for the above NLP problem are: If x* is an optimal sclution and a suitable "constraint qualification" holds at x* then the following condiions also hold:

(1) x* is feasible;

(2) there exist multipliers $\lambda_i \geq 0$, $i=1,\ldots,m$
and unconstrained multipliers λ_i, $i=m+1,\ldots,q$
such that: $\nabla f(x^*) + \sum_{i=1}^{q} \lambda_i g_i(x^*) = 0$;

and

(3) $\lambda_i g_i(x^*) = 0$ $\quad i=1,\ldots,m$

The K-T conditions are necessary but not generally sufficient criteria for identifying an optimal point. As such they are important in theoretical developments and serve as a basis for many computational algorithms. They also have the added attraction that they constitute a first order necessary condition for a minimum which can be tested numerically.

3.3 The Constrained Problem Transformed to an Unconstrained Problem [12,15]

The solution to the general NLP problem can often be found by replacing it with a sequence of unconstrained problems, each containing the constraints implicitly, whose limiting solution is the NLP problem solution. This unconstrained optimization (indirect) approach is motivated by several practical considerations. These are:

(1) The unconstrained formulations of the constrained problems are often simpler to apply or use;

(2) Many more efficient algorithms have been developed to handle unconstrained problems;

(3) The unconstrained algorithms have been more extensively studied and further developed;

(4) One avoids the necessity of having to cope individually with the boundaries of the feasible region.

The disadvantage of these methods is that they are not as efficient for some problems as the direct methods since the structure of the original problem is obscured. This is especially true when one has linear constraints or a linear objective function. The resulting solutions may also only be poor approximations to the correct solution instead of exact.

3.4 Penalty-Barrier Methods [20,44]

One way to modify the objective function in order to incorporate the influence of constraints is to add something to it which would bend the search direction into the

feasible region. In penalty (exterior point) methods one attempts to create an infinite penalty for departing far from the feasible region, and in barrier (interior point) methods one attempts to set up a barrier against leaving the feasible region.

The following are forms of the Penalty-Barrier methods:

Penalty-Constrained NLP Problem:

"Min $P(x,r)=f(x)+r_k^{-1}P(x)$

$\lim_{k\to\infty} r_k=0^+$

with respect to all $x\in E^n$."

Barrier-Constrained NLP problem:

"Min $P(x,r)=f(x)-r_kB(x)$

$\lim_{k\to\infty} r_k=0^+$

with respect to all $x\in\{X\}$"

3.4.1 Penalty Constrained (Exterior Point) Methods [12,20]

By construction, the penalty function $P(x)$ is ideally zero for $x\in X$, positive for $x\notin X$, and usually forces $F(x,r)$ toward $+\infty$ for $r\to 0$ and $x\notin X$. The typical penalty function strategy is to solve a sequence of penalty-constrained problems with r given successively smaller positive values.

Typically the penalty functions are defined for inequality constraints as:

$P_j(x)=0 \qquad$ for $g_j(x)\leq 0$

$\quad = g_j^2(x) \quad$ for $g_j(x)>0$

"Min f(x) such that a≤x≤b"

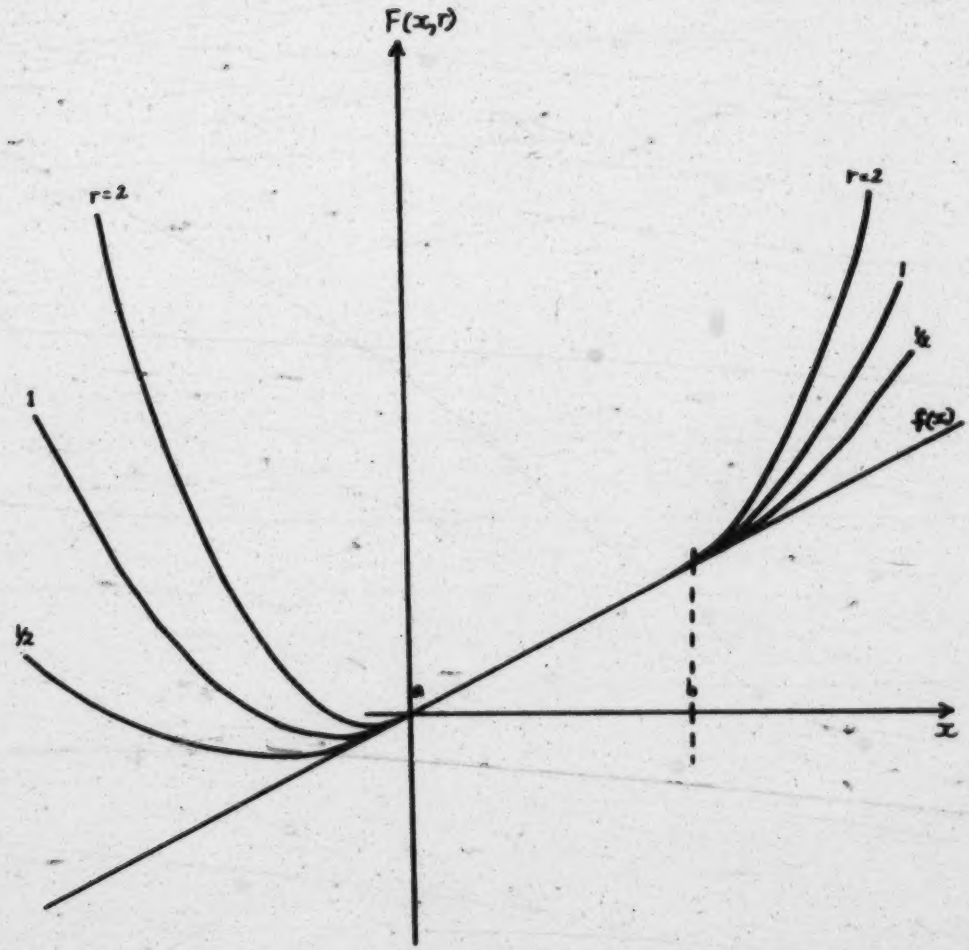

Figure 3.1 Penalty method

"Min f(x) such that a≤x≤b"

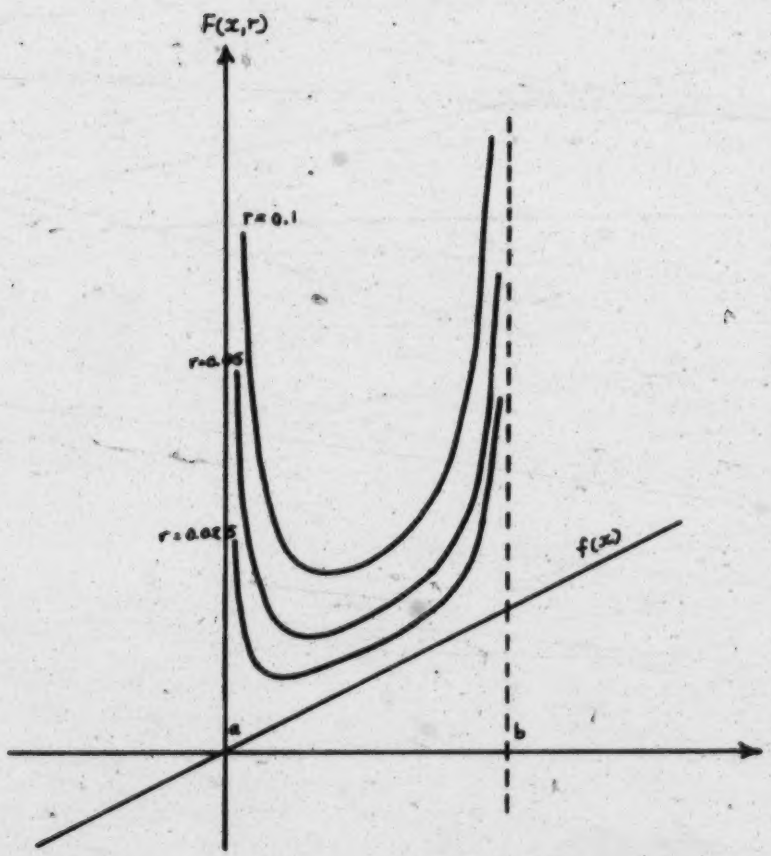

Figure 3.2 Barrier Method

This can be formulated as

$$P_i(x) = ((g_i(x) + |g_i(x)|)/2)^2$$

or as

$$P_i(x) = \text{Max}^2\{0, g_i(x)\}$$

For equality constraints

$$P_i(x) = h_i^2(x)$$

Thus, given the NLP problem

"Min $f(x)$

subject to $g(x) \leq 0$

$h(x) = 0$"

$$F(x,r) = f(x) + r\left(\sum_{i=1}^{m} g_i(x) + \sum_{i=m+1}^{q} h_i(x)\right)$$

The advantages of the penalty-constrained method are that the initial point x^0 does not have to lie within the feasible region and that it handles both equality and inequality constraints equally well.

The disadvantages of the method are:

(1) it does not produce intermediate solutions that are feasible;

(2) the problem becomes ill-conditioned as the penalty coefficient increases;

(3) it is often difficult to minimize as there is a lack in the orders of differentiability in x of the function at any boundary point of the feasible region;

(4) numerical errors in the penalty terms become significant for large penalty coefficients.

3.4.2 Barrier Constrained (Interior Point) Methods [12,20]

For $x \in \text{Int}(X)$ the barrier function $B(x)$ is continuous, tends towards $-\infty$ as any component of $g(x)$ tends to 0^-, and $\lim_{r_k \to \infty} r_k B(x)$ ranges on $(-\infty, 0]$. As with the penalty constrained problem, the strategy calls for solving a sequence of problems, in which the parameter r is assigned successively smaller positive values.

Typically $B(x)$ is defined as

$$(1) \quad B(x) = \sum_{i=1}^{m} g_i^{-1}(x)$$

or

$$(2) \quad B(x) = \sum_{i=1}^{m} \text{Ln}(-g_i(x))$$

Thus, the NLP problem

"Min $f(x)$

subject to $g(x) \leq 0$"

becomes

$$F(x,r) = f(x) - rB(x)$$

The advantages of the barrier constrained method are:

(1) it produces intermediate feasible sclutions;

(2) it gives a final solution of arbitrary accuracy.

The disadvantages of the method are:

(1) an initial point in the feasible region is required;

(2) the rapid change of $B(x,r)$ in the vicinity of the boundary complicates the one-dimensional optimization problem which most techniques require;

(3)at the boundary B(x,r) gets very large and $r_k \rightarrow 0^+$ so the function is very sensitive to variable changes and round-off error is intrcduced;

(4)it dces not handle equality constraints.

3.4.3 Combined Penalty-Barrier Problems [12]

There are many circumstances under which it is desireable to use a combined penalty-barrier algorithm. If one wishes to take advantage of the superior behavior of the interior point methods but also has equality constraints, then the mixed algorithm must be used. It is also necessary to use it in penalty constrained problems when continual satisfaction of certain constraints is necessary (such as when one of the constraints or the objective function contains a logarithmic term).

This problem takes the form

"Min $F(x,r)=f(x)+r^{-1}P(x)-rB(x)$"

In this method the initial point must lie within the feasible region defined by the constaints contained in B(x).

3.5 The Sequential Unconstrained Minimization Technique (SUMT) [12]

SUMT was developed by Fiacco and McCormick [9,10,11,12] based on an earlier technique proposed by C.W. Carrol. It is a mixed penalty-barrier technique which deals efficiently with non-linear objective functions and both equality and

inequality constraints. It generates a sequence of solutions which converges to an extremum.

The convexity or concavity of the functions involved is not critical to the performance of the algorithm; however if certain conditions are not satisfied it may converge to a local instead of a global extremum.

The following conditions are necessary for a local solution:

(1) the interior feasible region is not empty and an initial interior feasible point x^0 can be found;

(2) the functions $f(x)$ and $g(x)$ are twice continuously differentiable;

(3) $f(x)$ is bounded below on the feasible region;

(4) all local minima are located at finite points;

(5) if in addition $f(x)$ and $g(x)$ are convex and at least one is strictly convex, then the method will converge to the global minimum.

Given the problem

"Min $f(x)$

subject to $g(x) \leq 0$

$h(x) = 0$"

then the objective function is defined as

$$F(x,r) = f(x) - r\sum_{i=1}^{m} \ln(g_i(x)) + r^{-1}\sum_{i=m+1}^{q} h^2(x)$$

or $$F(x,r)=f(x)-r\sum_{i=1}^{m}(1/g_i(x))+r^{-1}\sum_{i=m+1}^{q}h_i^2(x)$$

Definition of Algorithm

(1) Given $x^0 \in X$ and $r_1>0$ set k=1;

(2) Minimize (using an unconstrained minimization technique):

$$F(x,r)=f(x)-r_k\sum_{i=1}^{m}\mathrm{Ln}(g_i(x))+r_k^{-1}\sum_{i=m+1}^{q}h_i^2(x);$$

(3) Set $r_{k+1}=r_k/c$ where $c>1$ is some constant and $k=k+1$

(4) Test for convergence:

If $$\left[\frac{\|F(x,r_k)-F(x,r_{k+1})\|}{\|F(x,r_{k+1})\|}\right] > \epsilon'$$

go to step 2; else stop.

Comments

(1) Other criteria, such as testing for the difference in magnitude between successive values of x, can be used for termination;

(2) The selection of r and c are often critical to the rate of convergence. Generally r should not be chosen too small and c should not be chosen to be too large or too small as otherwise convergence difficulties may be encountered. For more information on the choice of r and c see [22].

3.6 Rosen's Gradient Projection Algorithm

In this section a direct method for the solving of the constrained minimization problem is discussed. This method generates the search direction by projecting the negative

gradient $-\nabla f(x)$ cnto a subset of the constraints binding at x.

3.6.1 The Projection Matrix [24]

Given q linearly independent n-vectors $e_1, e_2, \ldots, e_q$ in E^n, these n-vectors span a q-dimensional subspace denoted by $\underline{Q}$. If q=n, then $\underline{Q}$ ccincides with E^n.

The equation $e_i^T x = c$ defines a hyperplane H_i in E^n or manifold of dimension n-1. The intersection H_1 to H_q of q of these hyperplanes creates an n-q dimensional subspace, denoted by Q. Thus Q is the set of all x with

$$(e_1, e_2, \ldots, e_q)^T x = E_q^T x = 0$$

Each of the vectors e_1 to e_q is perpendicular to Q, therefore Q and $\underline{Q}$ are mutually orthogonal. Together Q and $\underline{Q}$ span the space E^n.

Define the q×n matrix $\underline{P}$ [33] as constructed by the rule

$$\underline{P}_q = E_q^T (E_q E_q^T)^{-1}$$

The matrix $\underline{P}_q$ is a projection matrix which takes any vector in E^n into $\underline{Q}$ [33]. The n×n matrix P_q is now defined by

$$P_q = I - E_q^T (E_q E_q^T)^{-1} E_q$$

Thus P_q is a projection matrix which takes any vector in E^n into the intersection Q [33].

It can also be shown that

$$P_q^T = P_q \quad \text{and} \quad P_q^T P_q = P_q$$

In Figure 3.3 the projection z is thus obtained as $z=P_q x$.

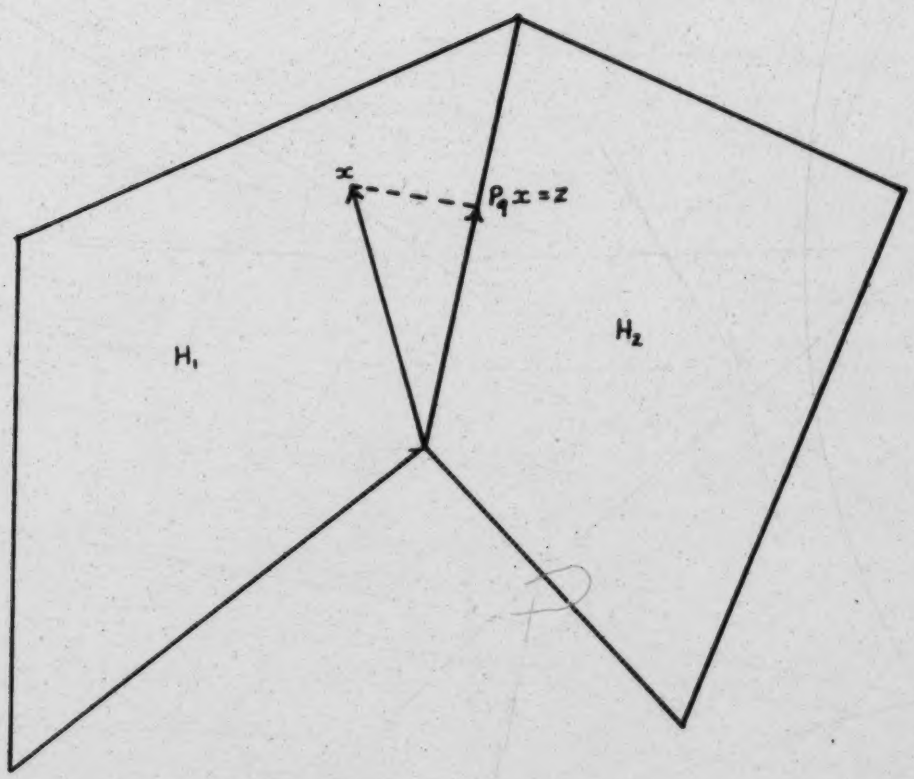

Figure 3.3 The projection $z=P_q x$

3.6.2 Rosen's Gradient Projection [33]

Let the objective function $f(x)$ be convex with continuous second partial derivatives in the feasible region X. The vector x is constrained by m linear inequalities of the form

$$A^T x - b \leq 0$$

where $A=[a_1, a_2 \ldots a_m]$

The constraints are normalized so that

$$\|a_i\| = 1 \qquad i=1,2,\ldots,m$$

These constraints restrict the solution to m closed half-spaces. Their intersection is, in general, a convex polytope and is called the feasible region X.

Let $A_q=[a_1, a_2 \ldots a_q]$ where $a_i=(a_{1i}, a_{2i}, \ldots a_{ni})$

$b_q=(b_1, b_2, \ldots, b_q)$

and $Q=[H_1, H_2 \ldots H_q]$

where Q is the set of hyperplanes active at x.

If the point x lies at the intersection of q linearly independent hyperplanes $H_1, H_2, \ldots, H_q$ which are defined by the q linearly independent vectors $a_1, a_2, \ldots a_q$ and the components $b_1, b_2, \ldots b_q$ $(q \leq m)$, then the q equations

$$A_q^T x - b_q = 0$$

determine the points that lie on the intersection of $H_1, H_2, \ldots, H_q$. Given A_q, the $n \times n$ symmetric projection matrix P_q is given by $P_q = I - A_q^T (A_q A_q^T)^{-1} A_q$

Thus P_q will take any vector in E^n into Q.

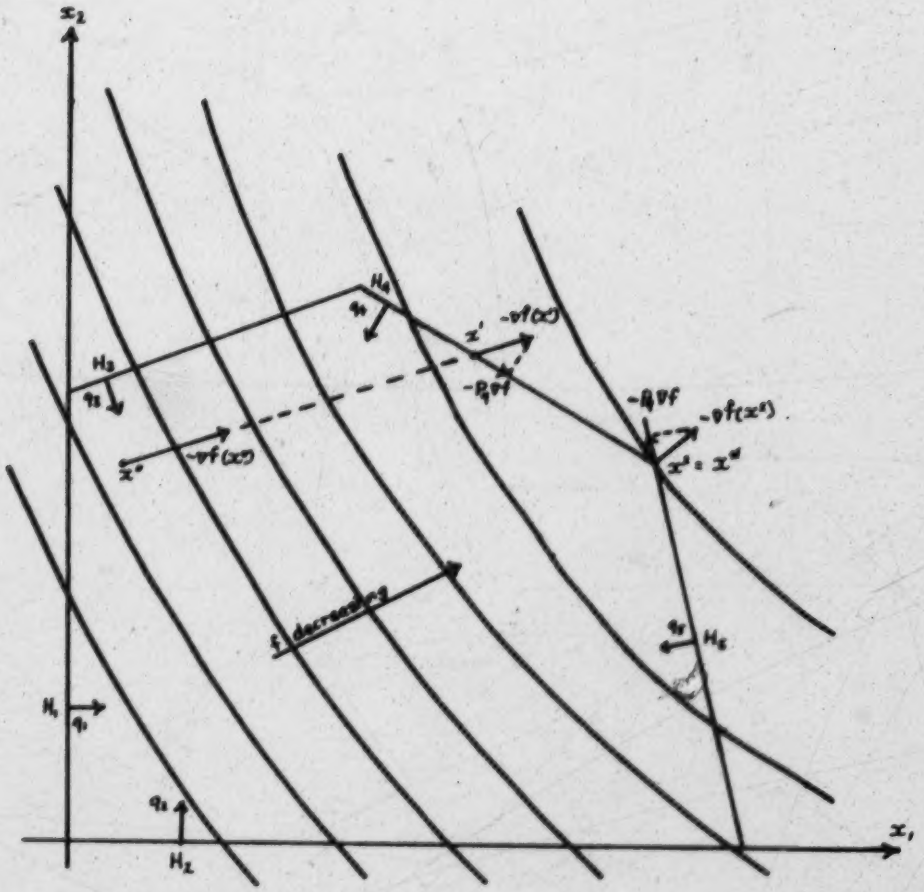

Figure 3.4 Gradient projection as a function of two variables

DEFINITION OF ALGORITHM

(1) Normalize the linear constraints

(3) Test the feasibility of the initial point x^0 and determine the constraints active there

(4) If some of the constraints are active then select those that form a linearly independent set and go to (5);
Else set P_q equal to the identity matrix and go to (7)

(5) Calculate $(A_q A_q^T)^{-1} \triangleq T$

(6) Calculate the projection matrix P_q

(7) Calculate the direction of search d (see 3.6.4)

(8) Calculate the vector R by

$$R = T_q A_q^T (-\nabla f)$$

and let R_t = the maximum element of R then

(9) If $\| d \| \leq \epsilon$ or $\| P_q (-\nabla f) \| \leq \epsilon$
Go to (10);
Else go to (11)

(10) If each element of R is ≤ 0 then a minimum has been found. If one or more elements of R are >0, then the objective function can be further minimized by dropping the hyperplane corresponding to R_t and returning to (5) [33]. If none of the above go to (11)

(11) Let Beta equal the maximum of the sums of the absolute values of the elements of the rows of T_q
If $R_t >$ Beta drop H_t from Q

(12) Normalize d and determine the maximum stepsize, τ_m, that can be taken along d without causing any of the constraints to be violated

(13) Let $y=x+\tau_m d$

If $\langle d, \nabla f(y)\rangle \geq 0$ set $x=y$ and go to (14);

Else interpolate to find the y that minimizes f along d and set $x=y$, then go to (5)

(14) Add to Q the hyperplane which corresponds to the stepsize τ_m and go to (5)

3.6.3 Calculation Requirements of Gradient Projection [33]

(1) Calculation of $(A_q A_q^T)^{-1}$ by the use of recursion relations:

(a) Dropping a hyperplane H_q:

Assume that $(A_q A_q^T)^{-1}$ is known and $A_{q-1} = [a_1\ a_2 \cdots a_{q-1}]$.

Suppose a square matrix C is partitioned so that

$$C=\begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix}$$

where C_1, C_4 are square matrices and C_2, C_3 are rectangular matrices. Then

$$C^{-1}=B=\begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$$

It can be shown that [33]:

$$C_1^{-1} = B_1 - B_2 B_4^{-1} B_3$$

Applying this to determining $(A_{q-1} A_{q-1}^T)^{-1}$, let

$$B=C^{-1}=(A_q A_q^T)^{-1}$$

and therefore B_1, B_2, B_3, and B_4 are known. In particular, B_1 is a $(q-1)\times(q-1)$ symmetric matrix, B_2 a $(q-1)$ column vector, $B_3 = B_2^T$, and B_4 a scalar.

If a hyperplane H_1 $(1<q)$ is to be dropped, the

relation still applies if the l-th and q-th rows and columns of B are interchanged before it is applied.

(b) Adding a hyperplane H_q:

Let $D_0 = C_4 - C_3 C_1^{-1} C_2$

and $D^{-1} = B = \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$

then $B_1 = D_1^{-1} + D_2^{-1} D_0 D^{-1} D_3 D_1$

$B_2 = -D_1^{-1} D_2 D_0^{-1}$

$B_3 = -D_0^{-1} D_3 D_1^{-1}$

$B_4 = D_0^{-1}$

If $D = C_q C_q^T$

and $D_1 = C_{q-1} C_{q-1}^T$

then $D_2 = C_{q-1} a_q = D_3^T$

and $D_4 = a_q^T a_q = 1$

giving $D_0 = a_q P_{q-1} a_q = \|P_{q-1} a_q\|^2 = \|a_q - C_{q-1} F\|^2$

This gives:

$B_1 = D_1^{-1} + D_0^{-1} F F^T$

$B_2 = -D_0^{-1} F = B_3^T$

$B_4 = D_0^{-1}$

where $F = D_1^{-1} D_2$

Thus allowing one to construct

$$B = (A_q A_q^T)^{-1}$$

(c) The first time $(A_q A_q^T)^{-1}$ is calculated it is done by Gaussian Elimination.

(2) Calculation of the projection matrix P_q:

(a) The first time P_q is calculated and everytime a hyperplane is dropped

$$P_q = I - A_q (A_q A_q^T)^{-1} A_q$$

(b) The recursion relation for the addition of a hyperplane is [33]:

$$P_{q+1} = P_q - P_q a_{q+1} a_{q+1}^T P_q / a_{q+1}^T P_q a_{q+1}$$

(c) Dropping a hyperplane from Q:

a. The point x^* is a constrained global minimum of $f(x)$ iff

$$-P_q \nabla f(x^*) = 0 \qquad (\|P_q \nabla f(x^*)\| \leq \epsilon)$$

and $$-(A_q A_q^T)^{-1} A_q \nabla f(x^*) = R \leq 0$$

If one or more components of R are positive, this indicates that the objective function can be decreased by dropping the hyperplane corresponding the the largest positive component of R, R_m.

b. If $\|P_q \nabla f(x)\| > \epsilon$, it may still be desirable to drop a hyperplane. Comparing the lengths of $-P_q \nabla f(y)$ and $-P_{q-1} \nabla f(y)$ in Figure 3.5 one can see that more is to be gained by moving along $-P_{q-1} \nabla f(y)$ than along $-P_q \nabla f(y)$, therefore H_1 should be dropped from Q. To determine if this should be done one calculates the sums of the absolute values of the elements of the rows of $(A_q A_q^T)^{-1}$ and sets R_m equal to the maximum of these sums. If $R_m > \beta$, then drop H_m from Q.

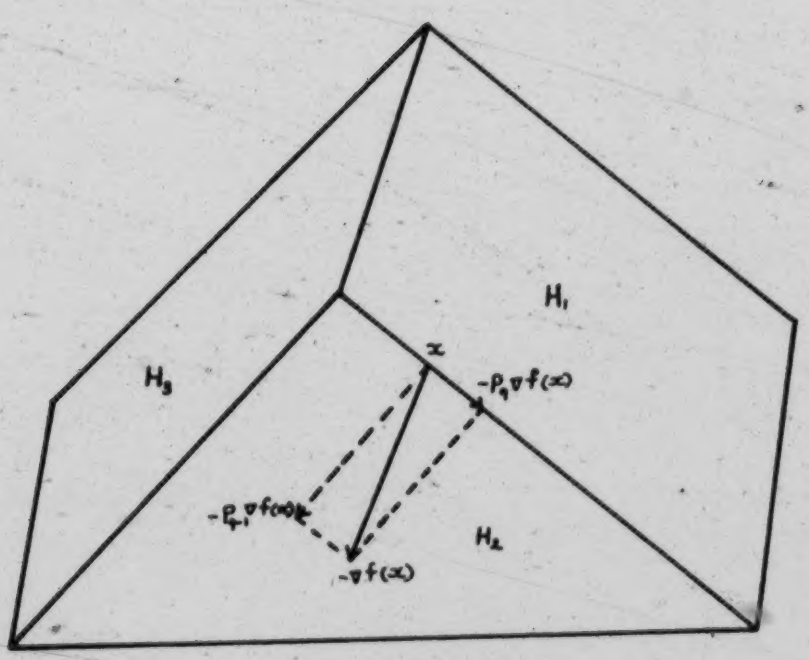

Figure 3.5 Dropping a hyperplane

(3) Calculation of the maximum stepsize τ_m

Given $y=x+\tau d$

we wish to find the maximum value of τ for which all constraints are satisfied. At the point where this line intersects the hyperplane H_i

$$y=x+\tau_i d$$
$$a_i^T y-b_i=0$$

substituting for y and solving for τ_i gives

$$\tau_i=(b_i-a_i^T x)/a_i^T d$$

τ_i is calculated for all hyperplanes not in Q. The minimum positive value of τ_m is the maximum stepsize which can be taken.

(4) Interpolation Routines:

If the stepsize is infinite then there is no constraint in the direction of d and cubic interpolation is used to find the minimum of f along d.

If the stepsize is finite then repeated linear interpolation is used until

$$\|d\nabla f(y)\|<\epsilon$$

where $y=x+\alpha d$ (α found by interpolation)

(cubic interpolation could be used here instead)

3.6.4 Calculation of the Search Direction d

(1) Pure gradient projection:

This is the method originally proposed by Rosen and consists of projecting the negative gradient onto the active constraint set. The method possesses the poor

convergence characteristics of the Steepest Descents algorithm and has only simplicity to recommend it. The direction of search is given by

$$d=-P_q \nabla f(x)$$

(2) PARTAN Acceleration:

Here the PARTAN acceleration method is adopted in order to improve the convergence of Rosen's algorithm. Only the Continued PARTAN algorithm is used to determine d. In it, every time a hyperplane is either added or dropped the acceleration routine is re-initialized. See the discussion of PARTAN in 2.3.

(3) Conjugate-Gradient Projection Acceleration:

Here the Fletcher-Reeves C-G algorithm is extended to Rosen's method. This extension was suggested by Goldfarb [16] and was implemented practically in this study. The directions of search are determined by

$$d^i=-P_q \nabla f(x^i)+\|P_q \nabla f(x^i)\|^2 d^{i-1}/\|P_q \nabla f(x^{i-1})\|^2$$

while the initial search direction is given by

$$d^0=-P_q \nabla f(x^0)$$

Every time a hyperplane is either added of dropped the acceleration routine must be re-initialized. It is also re-initialized after n iterations. As long as the constraint matrix remains unchanged, this method possesses the excellent convergence characteristics of the unconstrained C-G algorithm.

(4) Davidon-Fletcher-Powell Variable Metric:

Here the DFP algorithm as developed by Goldfarb [16] is

extended to Rosen's method.

The directions of search are given by

$$d=-H\nabla f(x)$$

In the first iteration H^0 is set equal to P_q.

Whenever a hyperplane is dropped H is updated by

$$H_{q-1}=H_q+P_{q-1}a_q a_q^T P_{q-1}/a_q^T P_{q-1} a_q$$

and whenever a hyperplane is added H is updated by

$$H_{q+1}=H_q-H_q a_{q+1}a_{q+1}^T H_q/a_{q+1}^T H_q a_{q+1}$$

When the constraint set remains constant H is updated by

$$H_q^{i+1}=H_q^i+B+C$$

where $z=\hat{t}d$

$$y=\nabla f(x^{i+1})-\nabla f(x^i)$$

$$B=zz^T/z^T y$$

$$C=-H_q^i yy^T H_q^i/y^T H_q^i y$$

This extension has been found to improve greatly the convergence characteristics of Rosen's algorithm.

3.7 Experimental Results

In this section the results of applying the methods of constrained minimization discussed in Chapter 3 to a set of constrained problems are presented. No claim of completeness or special difficulty is made for these problems; they are presented, in the main, as typical examples.

Experimental Conditions

In each case the functional f(x) is to be minimized, x^0 is the starting point and x^* is the optimum point which minimizes f(x) over the feasible set X.

Starting Conditions: Each test problem has its own starting point x^0. For the methods using the variable metric algorithms, the initial matrix H^0 is taken to be the identity matrix I.

Stopping Conditions: These vary with each method. See the discussion of experimental results for each problem. The computer code used in the sclution of these problems and a guide to its use may be found in [49,50].

3.8 Experimental Problems

The problems used to test the programs implementing the methods in Chapter 3 are given below. Each problem is given a number and is referred to by that number.

TP10: (Tabak and Kuo [38])

$$f(x) = cx_1 + x_3$$

$$g_1(x) = a_1 - x_1$$

$$g_2(x) = -a_2 + x_1$$

$$g_3(x) = d_1 - x_3$$

$$g_4(x) = -d_2 + x_3$$

$$g_5(x) = -w + x_4$$

$$g_6(x) = -q + x_5$$

$$g_7(x) = -x_1[x_1^2 - (2-4x_5^2) - x_4^2] + x_5$$

$$g_8(x) = x_1[(2-4x_5^2)(x_1x_4)^2 - x_4^4] - 2x_2x_3x_4x_5 + x_2 + 2x_3x_4$$

$-x_4^2 x_5$

$g_4(x) = -x_5$

$(a_1, a_2, d_1, d_2, c, w, q) = (0.1, 1.0, 0.5, 0.707, 0.010, 3.0, 10)$

$x^0 = (10.0, 0.5, 0.7, 1.0, 1.0)$

$x^* = (4.0, 1.0, 0.5, 0.5, 3.001)$

$f(x^*) = 0.540$

This NLP problem gives the solution to a third-order nonlinear system whose transfer function is

$$G(s,p) = 1/[(s+a)(s^2+2dws+w^2)]$$

$$p = (a, d, w) > 0$$

The requirement is to maximize the gain $K = 1/x$ (or minimize x) and still retain the stability of the system.

TP11: [20]

$f(x) = -2x_1^2 - x_1 x_2 + x_2^3$

$g_1(x) = x_1 + 3x_2^2 - 2$

$g_2(x) = 1 - 3x_1 + 2x_2$

$x^0 = (1.5, 0.2)$

$x^* = (1.995, 0.04134)$

$f(x^*) = -8.0414$

TP12: [20]

$f(x) = -x_1 x_2 + x_3 x_4$

$g_1(x) = x_1^2 + 2x_3^2 - 1$

$g_2(x) = x_2^2 - x_3 + 3x_4^2 - 1$

$x^0 = (0.8, 1.0, 0.1, -0.1)$

$x^0 = (-0.8, -0.5, 0.4, -0.2)$

$x^* = (-0.947, -1.096, 0.2245, -0.0864)$

$x^* = (0.947, 1.096, 0.2245, -0.0864)$

$f(x^*) = -1.058$

This problem has convex constraints but a non-convex objective function. Examination shows that there are, in addition to the origin, two other points at which the Kuhn-Tucker conditions are satisfied:

$x = (0, 0, -2/3, 1/3)$

$f(x) = -0.222$

and

$x = (0, 0, 0.707, -0.754)$

$f(x) = -0.533$

TP13: Colville's Test Problem One [22]

$$f(x) = -15x_1 - 27x_2 - 36x_3 - 18x_4 - 12x_5$$
$$+30x_1^2 - 40x_1x_2 - 20x_1x_3 + 64x_1x_4 - 20x_1x_5$$
$$+39x_2^2 - 12x_2x_3 - 62x_2x_4 + 64x_2x_5$$
$$+10x_3^2 - 12x_3x_4 - 20x_3x_5 + 39x_4^2 - 40x_4x_5 + 30x_5^2$$
$$+4x_1^3 + 8x_2^3 + 10x_3^3 + 6x_4^3 + 2x_5^3$$

subject to $Ax \leq b$ where

$$A = \begin{bmatrix} 16 & -2 & 0 & -1 & 0 \\ 0 & 2 & 0 & -0.4 & -2 \\ 3.5 & 0 & -2 & 0 & 0 \\ 0 & 2 & 0 & 4 & 1 \\ 0 & 9 & 2 & -1 & 2.8 \\ -2 & 0 & 4 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \\ -1 & -2 & -3 & -4 & -5 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

b = (40,2,0.25,4,4,1,40,60,-5,-1,0,0,0,0,0)

x^0 = (0,0,0,0,0)

x* = (0.3,0.33347,0.4,0.42831,0.22396)

f(x*) = -32.34868

3.9 Numerical Studies

The practical results of solving the constrained problem by transforming it into an unconstrained problem are discussed in sections 3.9.1 to 3.9.3 below. In section 3.9.4 Rosen's gradient projection algorithm is discussed and its performance on TP13 is compared to a solution of the same problem by the SUMT algorithm. The conclusions drawn from the numerical studies are presented in section 3.9.5.

3.9.1 The Penalty Method

The penalty method was applied to the solution of TP10, TP11, and TP12. DFP, SSVM, RVM, CG and PARTAN were applied to the solution of all three problems. The form of the method chosen was

"Min $P(x,r) = f(x) + r^{-1}P(x)$

Where $P(x) = \sum_{i=1}^{m} \max^2\{0, g_i(x)\}$"

for several different values of r. The results are given in Tables 3.1 to 3.3.

TEST	PROB	IPTAN	CPTAN	CG	DFP	SSVM	RVM
	IT	7*	7*	351*	351*	9*	351
TP10	#F	32	32	1075	730	38	730
	TIME	0.130	0.130	2.502	2.040	0.157	2.050
	IT	12	25	9*	6	6	6
TP11	#F	33	56	23	14	13	14
	TIME	0.085	0.113	0.075	0.072	0.071	0.070
	IT	23	23	16	11	17	10
TP12	#F	59	63	43	27	52	22
	TIME	0.111	0.117	0.108	0.117	0.155	0.107

* - METHOD DID NOT CONVERGE TO CORRECT SOLUTION

Table 3.1 Penalty solutions r=0.2

TEST	PROB	IPTAN	CPTAN	CG	DFP	SSVM	RVM
	IT	7*	7*	351*	20	13*	#
TP10	#F	42	42	1164	367	62	
	TIME	0.147	0.146	2.574	1.001	0.202	
	IT	13	19	12	20	16	35
TP11	#F	31	56	29	177	62	233
	TIME	0.087	0.107	0.082	0.201	0.120	0.256
	IT	24	23	21	#	#	#
TP12	#F	83	79	63			
	TIME	0.131	0.127	0.131			

* - METHOD DID NOT CONVERGE TO CORRECT SOLUTION
# - METHOD NOT STABLE

Table 3.2 Penalty solutions r=0.02

TEST	PROB	IPTAN	CPTAN	CG	DFP	SSVM	RVM
	IT	7*	7*	351*	39	17	#
TP10	#F	51	51	1489	229	92	
	TIME	0.243	0.247	3.038	0.569	0.262	
	IT	23	19	21	151*	151*	32
TP11	#F	73	59	57	484	405	230
	TIME	0.179	0.113	0.107	0.609	0.562	0.252
	IT	21	21	99	#	151*	#
TP12	#F	70	68	326		378	
	TIME	0.117	0.112	0.421		0.864	

* - METHOD DID NOT CONVERGE TO CORRECT SOLUTION
\# - METHOD NOT STABLE

Table 3.3 Penalty solutions r=0.002

Comments

Generally the solutions were inaccurate with the best solutions having errors in x of approximately 10%. For several of the problems the solutions were so inaccurate as to be completely unacceptable while others converged to local minima. As expected, a decrease in the value of r usually resulted in an increase in both accuracy and the number of iterations required to solve the problem. The variable metric algorithms experienced some difficulty in obtaining solutions and in some problems the algorithm itself failed resulting in no solution. This, however, could be corrected by restarting the searches at suitable intervals.

3.9.2 The Barrier Method

The barrier method was applied to the solution of TP10, TP11, and TP12. DPP, SSVM, RVM, CG and PARTAN were applied to the solution of all three problems. The form of the method chosen was

"Min $F(x,r)=f(x)-rB(x)$

Where $B(x)=\sum_{i=1}^{m} 1/g_i(x)$"

for several different values of r. The results are given in Tables 3.4 to 3.6.

TEST	PROB	IPTAN	CPTAN	CG	DFP	SSVM	RVM
	IT	41*	77*	80*	23*	37*	26*
TP10	#F	157	268	287	88	143	93
	TIME	0.667	0.719	0.731	0.320	0.508	0.337
	IT	12*	16*	10*	10	8	10
TP11	#F	45	56	29	30	30	29
	TIME	0.075	0.059	0.080	0.088	0.086	0.089
	IT	25*	15*	10*	11	8	16
TP12	#F	81	62	27	33	22	50
	TIME	0.145	0.120	0.090	0.119	0.101	0.143

* - METHOD DID NOT CONVERGE TO CORRECT SOLUTION

Table 3.4 Barrier solutions $r=10^{-1}$

TEST	PROB	IPTAN	CPTAN	CG	DFP	SSVM	RVM
	IT	146*	106*	85*	29*	39*	34
TP10	#F	492	360	284	100	159	126
	TIME	1.222	0.897	0.722	0.287	0.553	0.440
	IT	13*	17*	20	151*	151*	151*
TP11	#F	50	62	71	997	1036	1013
	TIME	0.079	0.090	0.114	0.752	0.831	0.792
	IT	29*	34*	14*	12	13	11
TP12	#F	91	130	44	36	34	31
	TIME	0.156	0.191	0.108	0.125	0.125	0.117

* - METHOD DID NOT CONVERGE TO CORRECT SOLUTION

Table 3.5 Barrier solutions $r=10^{-2}$

TEST	PROB	IPTAN	CPTAN	CG	DFP	SSVM	RVM
	IT	124*	160*	199*	54	79	53
TP10	#F	401	520	663	232	330	190
	TIME	0.994	1.300	1.620	0.706	1.093	0.644
	IT	27*	23*	25	140*	151*	151*
TP11	#F	121	109	106	994	1078	960
	TIME	0.142	0.124	0.138	0.754	0.824	0.779
	IT	29*	48*	26	10	16	13
TP12	#F	102	171	95	41	62	41
	TIME	0.169	0.235	0.156	0.119	0.156	0.125

* - METHOD DID NOT CONVERGE TO CORRECT SOLUTION

Table 3.6 Barrier solutions $r=10^{-3}$

Comments

As with the penalty method, the solutions were inaccurate. Here too accuracy and number of iterations generally increased as r decreased in value. It can be seen that PARTAN is completely unsuitable for use as a minimization strategy for barrier problems solved in the manner presented here as the function does not sufficiently approximate a quadratic. Again the variable metric algorithms experienced some difficulty in finding solutions but not as much as with the exterior penalty method. Here also this difficulty could be corrected by restarting the searches at suitable intervals.

3.9.3 The SUMT Algorithm

The SUMT algorithm was applied to the solution of TP10, TP11, and TP12. DFP, SSVM, RVM, CG and PARTAN were applied to the solution of all three problems. The form of the method chosen was

"Min $P(x,r)=f(x)-rB(x)$

Where $B(x)=\sum_{i=1}^{m}\mathrm{Ln}(g_i(x))$"

and $r^0=1$ and $c=5$. (The algorithm implemented here for this study was the basic SUMT algorithm. No attempt was made to check for and include only active constraints as the logic required both for the main program and the function gradient subroutine is very complex and the results were not deemed

worth the effort.)

In addition the following were also studied:

(1) the relative effectiveness of the Golden and cubic linear search techniques;

(2) the advantage gained when using the variable metric algorithms of setting H^0 equal to the H generated by the previous unconstrained minimization instead of restarting the minimization by setting H^0 equal to the identity matrix.

The results of these studies are given in Tables 3.7 to 3.11.

TEST	PROB	IPTAN	CPTAN	CG	DFP	SSVM	RVM
TP10	IT	#	#	#	110	149	106
	#F				476	614	447
	TIME				2.098	2.793	1.911
TP11	IT	40	36	185	29	25	23
	#F	224	226	1122	169	151	146
	TIME	0.299	0.285	0.990	0.276	0.249	0.242
TP12	IT	100	115	822*	70	47	107
	#F	465	523	8412	314	232	412
	TIME	0.643	0.723	6.000	0.637	0.498	0.853

* - METHOD DID NOT CONVERGE TO CORRECT SOLUTION
\# - METHOD NOT STABLE

Table 3.7 SUMT solutions: Cubic interpolation

		RESTARTED			NOT RESTARTED		
TEST	PROB	DFP	SSVM	RVM	DFP	SSVM	RVM
TP10	IT	254	482	239	110	149	106
	#F	1142	2655	1050	476	614	447
	TIME	4.853	11.266	4.179	2.098	2.793	1.911
TP11	IT	40	30	37	29	25	23
	#F	238	203	231	169	151	146
	TIME	0.328	0.287	0.317	0.276	0.249	0.242
TP12	IT	209	149	#	70	47	107
	#F	926	777		314	232	412
	TIME	1.625	1.306		0.637	0.498	0.853

* - METHOD DID NOT CONVERGE TO CORRECT SOLUTION
\# - METHOD NOT STABLE

Table 3.8 SUMT(cubic): Effect of restarting

TEST	PROB	IPTAN	CPTAN	CG	DFP	SSVM	RVM
TP10	IT	959*	761*	608*	94	138	121
	#F	21427	16703	22099	1744	2562	4009
	TIME	46.364	35.265	36.514	4.465	6.735	9.872
TP11	IT	23	23	26	19	21	17
	#F	875	878	1004	450	528	420
	TIME	0.656	0.659	0.758	0.437	0.496	0.413
TP12	IT	89	61*	73	200	70	61
	#F	2168	1626	1973	5007	1345	1695
	TIME	1.675	1.244	1.523	4.661	1.357	1.577

* - METHOD DID NOT CONVERGE TO CORRECT SOLUTION

Table 3.9 SUMT solutions: Golden Search

		RESTARTED			NOT RESTARTED		
TEST	PROB	DFP	SSVM	RVM	DFP	SSVM	RVM
TP10	IT	163	517	185	94	138	121
	#F	3349	13909	7349	1744	2562	4009
	TIME	8.169	35.205	17.113	4.465	6.735	9.872
TP11	IT	19	22	19	19	21	17
	#F	600	655	599	450	528	420
	TIME	0.498	0.548	0.495	0.437	0.496	0.413
TP12	IT	172*	44*	93*	200	70	61
	#F	4197	984	3090	5007	1345	1695
	TIME	3.591	0.966	2.463	4.661	1.357	1.577

* - METHOD DID NOT CONVERGE TO CORRECT SOLUTION

Table 3.10 SUMT(Golden): Effect of restarting

TEST	PROB	SD	CPTAN	CG	DFP
GRAD	IT	14	12	11	20
PROJ	#F	15	11	10	10
	TIME	0.642	0.593	0.561	0.868
	IT	#	138*	561*	81
SUMT	#F		681	2698	467
	TIME		3.669	12.144	2.446

* - METHOD DID NOT CONVERGE TO CORRECT SOLUTION
\# - METHOD NOT STABLE

Table 3.11 Test problem 13

Comments

Experiments were also conducted in measuring the effects of varying the values of ϵ, ϵ' used to determine the convergence of the SUMT algorithm and in varying the value of c used. A standard of $\epsilon, \epsilon' = 10^{-4}$ was employed. Choosing a smaller value generally resulted in an increase in iterations with very little increase in accuracy. A larger value sometime resulted in poor solutions so 10^{-4} was selected as the best value to used on the problems being studied. A value of c=5 was chosen as giving the best rate of convergence with the fewest difficulties. A choice of c=3 resulted in no improvements and poorer performance for some examples. A larger c created difficulties during the initial SUMT iterations. When the minimum was sufficiently approached a strategy of increasing c to 10 was employed and this gave improved results with no detrimental effects.

As can be seen from Tables 3.7 and 3.9 the Golden Search has few advantages to recommend it over the cubic search. The conjugate gradients algorithm displayed somewhat superior performance when employing the Golden Search but for the other algorithms even though there was usually a decrease in the total number of iterations, the number of function evaluations was prohibitively large.

The cubic interpolation method as implemented presented no difficulties during the linear search. Some difficulty

is often experienced with the cubic interpolation algorithm when applied to barrier constrained problems. This is due to the fact that the linear search often finds a minimum in the infeasible region. However by checking for the feasibility of the point generated and by taking suitable action when it is infeasible, the method is readily adapted to the solution of these problems. Its performance is generally so superior to that of the Golden Search that it virtually precludes the use of that technique except under certain circumstances (see section 2.7.1 and 2.7.2).

The effect of retaining the old H instead of setting it equal to the identity matrix at the beginning of each SUMT iteration is dramatic. Not only is the rate of convergence 2 to 4 times faster but sometimes a solution is found where reinitializing the metric H results in no solution. Therefore, in general, the metric H should not be reinitialized unless during the solution of some particular problem the accumulative error results in H becoming either singular or inaccurate.

3.9.4 Rosen's Gradient Projection Method

Rosen's GP was applied to the solution of TP13. The SD, PARTAN, CG and DFP implementations of the algorithm were applied to the solution of this problem. In addition the SUMT algorithm was used to solve this problem in order to provide a basis for comparing the two different techniques.

The results are given in Table 3.11.

Comments

All the implementations of the GP algorithm provided accurate solutions with a small execution time. For a relatively well-behaved objective function such as in TP13, CG and PARTAN performed the best with DFP possessing the largest execution time and largest number of iterations. The larger executon time is due to the more complicated algorithm while the much larger number of iterations is due to the particular path selected and is peculiar to the problem and not typical. The SUMT algorithm behaved very poorly on this problem. The PARTAN and CG algorithms generated solutions but they were inaccurate with more than 10% error in the value of x^* and $f(x^*)$. Only the DFP algorithm found an accurate solution in a reasonable amount of execution time. In addition the DFP algorithm had to be restarted at each SUMT iteration; otherwise the inaccurate solutions of the first three SUMT iterations contaminated the metric H so much that it was unable to converge to the true solution.

The advantage of a direct method such as GP in solving constrained minimization problems over an indirect method such as SUMT was also apparent: GP displaying much faster convergence with fewer function evaluations and a smaller execution time.

3.9.5 Conclusions

The simple penalty and barrier methods are capable of producing approximate solutions and are simple to implement. However, some care must be exercised in choosing a minimization technique as some may perform well on a particular problem and others may not converge at all.

While the execution time of the SUMT algorithm is much higher than that of the true penalty and barrier methods, it is much more likely to converge to the solution and the solution produced is much more accurate.

In circumstances where it can be used, a direct method of solving constrained problems such as gradient projection likely has superior performance characteristics over indirect methods employing penalty/barrier concepts such as SUMT.

CHAPTER IV

4. Organizing the Optimal Control Problem for Mathematical Programming Solution

In this chapter the methods of mathematical programming will be applied to the optimal control of selected dynamic systems. In order that these optimal control problems can be solved, specific formulations of the problems are required which will be developed in the following sections. These will serve as the basis for further discussion.

4.1 Formulation of the Optimal Control Problem

In formulating the optimal control problem we will deal first with continuous-time dynamic systems. From these we will consider a discrete approximation enabling a solution by computer techniques. The dynamic behavior of each system will be described by a set of ordinary non-linear first-order differential equations of the form

$$\dot{x}=f(x,u,t)$$

where:

$x \in E^n$ is an n-dimensional state vector;

$u \in E^m$ is an m-dimensional control vector;

t = time;

$f(x,u,t) \in E^n$ is an n-dimensional non-linear vector function;

$\dot{x}=dx/dt$

In addition, one will be given the boundary values for the

system such as the initial and/or final state.

The control action consists of bringing the system from some initial state $x(t_0)$ to some final state $x(t_f)$ in such a manner that a specified performance index is minimized without violating the dynamic equations or any constraints. The general form of this performance index is

$$J(x,u,t) = \int_{t_0}^{t_f} L(x,u,t)\,dt + \phi(x(t_f))$$

In many practical applications constraints are imposed on the state and control vectors. These constraints are assumed to be of form

$$g(x,u) \leq 0$$

$$h(x,u) = 0$$

Some of the most commonly used cost functionals are:

(a) Minimum time: $J = \int_{t_0}^{t_f} dt = t_f - t_0$

(b) Minimum effort: $J = \int_{t_0}^{t_f} \| u(t) \|\, dt$

(c) Minimum energy: $J = \int_{t_0}^{t_f} u^2(t)\,dt$

4.2 Solution of the Optimum Control Problem

The main theoretical approaches to optimal control synthesis are based on the Calculus of Variations [17] and its derivatives: Pontryagin's Maximum Principle [32], and Dynamic Programming [4].

4.2.1 Calculus of Variations [17]

The classical calculus of variations, developed in the seventeenth century, can be used to solve the optimal control problem:

"Minimize $J=\Phi(x(t_f),t_f)+\int_{t_o}^{t_f} L(x(t),u(t),t)\,dt$

subject to $\dot{x}=f(x(t),u(t),t)$"

with $x(t_o)$ and/or $x(t_f)$ constrained.

Define the Hamiltonian H as

$$H(x(t),u(t),\lambda(t),t)=L(x(t),u(t),t)+\lambda^T(t)\,f(x(t),u(t),t)$$

Then standard theory gives the Euler-Lagrange equations

$$\dot{\lambda}^T=-\frac{\partial H}{\partial x}=-\frac{\partial L}{\partial x}-\lambda^T\frac{\partial f}{\partial x}$$

$$\lambda(t_f)=\frac{\partial \Phi}{\partial x}$$

$$\frac{\partial H}{\partial u}=0$$

In solving these equations one has less than a full set of boundary conditions specified at each end-point, resulting in a two-point boundary value problem (TPBVP) which is generally difficult to solve.

Pontryagin's minimum principle [32] consists of a set of locally necessary conditions for optimality. This involves the use of variables called costate or adjoint-system variables, (λ_i, i=1,...,n), similar to the Lagrange

multipliers.

There exists an adjoint vector $\lambda(t)$ such that:

$$\dot{x}^*(t) = \frac{\partial H}{\partial \lambda^*(t)}$$

$$\dot{\lambda}(t) = -\frac{\partial H}{\partial x^*(t)}$$

If $u^*(t)$ is the optimal control and $x^*(t)$ the optimal trajectory then

$$H(x^*, u^*, \lambda^*, t) \leq H(x^*, u, \lambda^*, t)$$

This condition replaces $\partial H/\partial u = 0$ for $u \in U$. This also results in a TPBVP.

4.2.2 Dynamic Programming [4]

Dynamic programming is based on the observation that "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision" (Bellman's principle of optimality). The method of dynamic programming was first developed by Bellman for discrete systems, and in the continuous form by Hamilton, Jacobi, and Bellman.

The major computational disadvantage of dynamic programming is that its recursive relations require an excessive amount of computer memory though this requirement has been partially reduced by other authors.

4.3 Discrete Formulation of the Continuous Optimal Control Problem [23,38]

In one method of formulating the optimal control problem as a mathematical programming problem, the continuous control problem is represented as a discrete control problem. The continuous optimal control problem is taken to be:

"Find the control $u^*(t) \in U$ such that:

$$\dot{x}(t) = f(x(t), u(t), t)$$

with boundary conditions (not all fixed)

$$x(t_0) = x_0, \quad x(t_f) = x_f,$$

follows the trajectory $x^*(t) \in X$ and minimizes the cost function

$$J(x,u,t) = \Phi(x(t_f)) + \int_{t_0}^{t_f} L(x(t), u(t), t)\, dt"$$

First the time interval $[t_0, t_f]$ is partitioned into N intervals, not necessarily all equal. The time points under consideration become $t_0, t_1, \ldots, t_N$ where $t_N = t_f$. We must now approximate the state equations by difference equations. In doing so, we shall use the simplest approximating difference equation:

$$x(t_k + \Delta t_k) \cong x(t_k) + f(x(t_k), u(t_k), t_k)\Delta t_k$$

For brevity $x(t_k)$ is written as $x(k)$. Then each state can be expressed as

$$x(k+1) \cong x(k) + f(x(k), u(k); k)\Delta t_k \quad \forall k \in [0, N-1]$$

Thus the control vector $u(t)$ becomes a sequence

$\{u(0), u(1), \ldots, u(N-1)\}$ and the state vector x is a sequence $\{x(0), x(1), \ldots, x(N)\}$, $\lim_{N} \to \infty$.

The integral cost functional can be represented as the limit of a summation. The system then becomes

"Minimize

$$J(x,u,k) = \lim_{N\to\infty}\left[\Phi(x(N)) + \sum_{K=0}^{N-1} L(x(k),u(k),k)\Delta t_K\right]$$

subject to $\lim_{\Delta t_K \to 0} 1/\Delta t_K (x(k+1) - x(k)) = f(x(k),u(k),k)$

$$\left.\begin{aligned} g(x(k),u(k)) &\leq 0 \\ h(x(k),u(k)) &= 0 \end{aligned}\right\} \begin{aligned} x(k) &\ X \\ u(k) &\ U \end{aligned}$$

where $\Delta t_K = t_{K+1} - t_K$

and $x(t_0) = x_0, \quad x(t_f) = x_f$"

Thus the continucus optimal control problem is a mathematical programming problem of infinite dimension. However in practical applications the number of variables must of necessity be finite and this number must be small enough that computer implementation becomes practical. The above problem is then refcrmulated as

"Minimize $\quad J(x,u,k) = \Phi(x(N)) + \sum_{K=0}^{N-1} L(x(k),u(k),k)\Delta t_K$

subject to $\quad x(k+1) = x(k) + \Delta t_K f(x(k),u(k),k)$

$$g(x(k),u(k)) \leq 0$$

$$h(x(k),u(k)) = 0$$

where $\sum_{K=0}^{N-1} \Delta t = t_f - t_0 = T$

and $x(0)=x_o$, $x(N)=x_f$"

While other more elaborate approximations of the integral can be used, this formulation is the simplest and most convenient to use.

The problem can now be solved using mathematical programming techniques such as penalty methods and Rosen's gradient projection.

If Rosen's gradient projection method is used and the state difference equations are non-linear then a technique employed in the method of quasilinearization must be used: The state equations are linearized about a nominal state-control history and then a sequence of linearized problems is solved from an initial state-control history (usually guessed) [48]. However in this study, for simplicity only linear systems are solved by Rosen's method and for further ease of programming all time intervals Δt are considered equal. The formulation given above will be applied to the solution of a sample problem in Chapter 5.

Programming Considerations

When solving the optimal control problem as a mathematical programming problem, in the format given above each component of the state or control vector is a separate variable at each discrete time point t_k and in many problems the time points themselves are variables. This gives N time variables, n(N+1) state variables, and mN control variables

for a total of $N(1+n+m)+n$ variables. If any boundary conditions such as $x(0)$ or $x(N)$ are given then the number of variables can be reduced accordingly. It should be kept in mind always that the above formulation is an approximation and the results should be interpreted appropriately. In some circumstances, such as high-frequency oscillatory systems this approximation will not work. It should be noted also that more elaborate difference schemes may be used to improve the accuracy of the solutions.

4.4 Balakrishnan's Epsilon Technique [2,3]

The epsilon method is another approach to the optimization of dynamic systems which avoids the explicit solution of the dynamic equations.

Given the problem

"Minimize $\int_{t_0}^{t_f} L(x,u,t)\,dt$

subject to $\dot{x}=f(x,u,t)$

given $x(t_0)=x_0$ and $x(t_f)=x_f$"

the epsilon method seeks to minimize the functional

$$J(x,u,t,\epsilon)=1/2\epsilon\int_{t_0}^{t_f}\|\dot{x}-f(x,u,t)\|^2dt+\int_{t_0}^{t_f}L(x,u,t)\,dt$$

as $\epsilon\to 0$.

An iterative approach is used in which the boundary conditions are always exactly satisfied but where a dynamic

error term (the norm term) in the system dynamics exists. This error term is reduced with each iteration until the system dynamics are satisfied. The method embodies some of the features of both parameter optimization by differential approximation and parameter optimization by the penalty function method. The following are the three essential points of the epsilon method as given by Taylor and Constantinides [39] from Balakrishnan [2,3]:

(1) The epsilon technique offers a non-dynamic formulation. The dynamic equations of the system are not explicitly solved;

(2) As the epsilon parameter approaches zero, the epsilon method yields a statement of Pontryagin's minimum principle. Thus the above functional provides a sequence of trajectories and controls that can be made arbitrarily close to the true optimum as epsilon approaches zero;

(3) Not only is $\lim_{\epsilon \to 0} J(x,u,t,\ )$ likely to have a solution but the epsilon formulation may have a solution even when the solution of the original problem does not exist.

In order to represent the epsilon functional so that $\dot{x}(t)$ is not identically $f(x,u,t)$, use is made of the Rayleigh-Ritz method. Also, since the Rayleigh-Ritz method is a method of parameter optimization, the functional minimization originally required becomes an ordinary minimization.

The Rayleigh-Ritz Expansion [3,39]

The method as used here consists of expanding the state variables and control variables in terms of an infinite sequence of functions

$P_i(t)$, $i=0,\ldots,\infty$

So that $x_j = p_o(t) + \sum_{i=1}^{\infty} a_i P_i(t)$ $\quad j=1,\ldots,n$

where $P_o(t)$ satisfies any boundary conditions and where the $P_i(t)$ vanish at the initial and terminal times.

Thus the cost is a function of the variables a_i which are determined through ordinary mathematical programming techniques. This technique can also reduce the total number of variables used in the problem. In this study the functions $(\sin(i\pi t/t_f))$ are chosen as the $P_i(t)$.

Other Ritz formulations can be used. These would allow the use of Chebyshev polynomials, cosines, etc., in place of the sine functions used. However, because of the modifications required in the programs and because of a lack of time and resources their implementation is left as a subject for further study.

Balakrishnan's epsilon method is applied to the solution of a test problem in Chapter 5.

CHAPTER V

5. Case Studies

The purpose of this chapter is to apply the mathematical programming techniques discussed in chapters 2, 3, and 4 to the solution of two selected control problems. The examples presented were chosen to demonstrate these techniques and as such are relatively simple and straight forward. Each problem is solved by more than one approach in order to illustrate and compare methods.

5.1 The Minimum Energy Regulator [19]

The first sample control problem is a linear system with a quadratic cost function and initial and final states specified. It will be referred to as the minimum energy regulator (MER) problem. It consists of:

"Minimize $J = \int_{t_o}^{t_f} u^2(t)\,dt$

subject to $\dot{x}_1 = x_2$

$\dot{x}_2 = u$

with the boundary conditions

$x_1(0) = -1.0$ $\qquad$ $x_1(t_f) = 0.0$

$x_2(0) = 0.0$ $\qquad$ $x_2(t_f) = 0.0$

$t_o = 0.0$ $\qquad$ $t_f = 2.20$

plus the additional constraints

$g(x,t) = x_1 + 3x_2 + 0.02t - 1.5 \leq 0$

$-1 \leq u \leq 1$

The analytical solution for this problem without g(x) can be easily found [19], but otherwise a computer solution is required.

In order to solve this problem using mathematical programming techniques it is discretized in the following manner. Divide $[t_o, t_f]$ into m equal intervals of length Δt. The problem is then reformulated as:

"Minimize $J(x,u,t) = \Delta t \sum_{i=0}^{m-1} u^2(i)$

subject to $x_1(i+1) = x_1(i) + \Delta t x_2(i) \quad i=0,1,\ldots,m-1$

$x_2(i+1) = x_2(i) + \Delta t u(i) \quad i=0,1,\ldots,m-1$

$g(x(i), t(i)) = x_1(i) + 3x_2(i) + 0.02t(i) - 1.5 \leq 0$

$-1 \leq u(i) \leq 1$

where $\Delta t = t_f/m$

$t(i) = i\Delta t$"

Let $t(i) = x_3(i)$

The above MP problem now has $4m+3$ variables and $5m+3$ constraints. For large m this would be expensive to solve. Using the difference equations and boundary conditions we can solve for the x(i) explicitly in terms of the u(i). This results in:

$$x_1(i) = x_1(0) + \Delta t \sum_{j=0}^{i-1} x_2(j) \quad i=1,\ldots,m$$

$$x_2(i) = x_2(0) + \Delta t \sum_{j=0}^{i-1} u(j) \quad i=1,\ldots,m$$

$$x_3(i) = x_3(0) + i\Delta t$$

Substituting for the $x_2(1)$ in the equation for $x_1(i)$ yields

$$x_1(1) = x_1(0) + \Delta t x_2(0)$$

$$x_1(i) = x_1(0) + i\Delta t x_2(0) + \Delta t^2 \sum_{j=1}^{i-1} \sum_{k=0}^{j-1} u(k) \qquad i=2,\ldots,m$$

$$= x_1(0) + i\Delta t x_2(0) + \Delta t^2 \sum_{k=0}^{i-1} (i-k-1)\, u(k) \qquad i=1,\ldots,m$$

Substituting into $g(x(i)) \leq 0$ yields

$$x_1(0) + i\Delta t x_2(0) + \Delta t^2 \sum_{k=0}^{i-1} (i-k-1)\, u(k) + 3\left(x_2(0) + \Delta t \sum_{k=0}^{i-1} u(k)\right)$$

$$+0.02(x_3(0) + i\Delta t) - 1.5 \leq 0 \qquad i=1,\ldots,m$$

Regrouping terms and substituting the BC's, then,

$$g(x(i)) = \sum_{k=0}^{i-1} [\Delta t^2 (i-k-1) + 3\Delta t]\, u(k) + 0.02(i\Delta t) - 2.5 \leq 0$$

$$i=1,\ldots,m$$

The equalities $x_1(m)=0$ and $x_2(m)=0$ reduce to $-x_1(m) \leq 0$ and $x_2(m) \leq 0$ as the equalities will hold if the energy is minimized. The inequalities

$$-x_1(m) \leq 0 \text{ and } -x_2(m) \leq 0$$

can be expressed as

$$-\Delta t^2 \sum_{k=0}^{m-2} (m-k-1)\, u(k) + 1 \leq 0$$

and $$\Delta t \sum_{k=0}^{m-1} u(k) \leq 0$$

Thus the problem has been reduced to an MP problem of m variables with $3m+2$ constraints. The reduced problem is

restated briefly:

"Minimize $J=\Delta t\sum_{i=0}^{m-1}u(i)^2$

subject to $u(i)-1\leq 0 \qquad i=0,1,\ldots,m-1$

$-u(i)-1\leq 0 \qquad i=0,1,\ldots,m-1$

$$\sum_{k=0}^{i-1}[\Delta t^2(i-k-1)+3\Delta t]u(k)+0.02(i\Delta t)-2.5\leq 0 \qquad i=2,\ldots,m$$

$$-\Delta t^2\sum_{k=0}^{m-2}(m-k-1)u(k)+1\leq 0$$

$$\Delta t\sum_{k=0}^{m-1}u(k)\leq 0"$$

This problem is now in a form that will allow it to be solved directly by programs implementing methods such as Rosen's Gradient Projection method or Fiacco and McCormick's SUMT.

5.1.1 Solution Using Rosen's Gradient Projection Method

Since the program does not contain a procedure for generating a feasible point from a non-feasible one and because of the difficulty in choosing a feasible point for non-trivial m, a slack variable, S, was introduced into the constraints:

If $u(i)=0.5 \qquad i=0,1,\ldots,[m/2]-1$

$u(i)=-0.5 \qquad i=m/2,\ldots,m-1$

then all the constraints are satisfied except

$$-\Delta t^2 \sum_{k=0}^{m-2} (m-k-1)\, u(k) + 1 \leq 0$$

If the slack variable S, with $S^0=1$, is added to it, it becomes

$$-\Delta t^2 \sum_{k=0}^{m-2} (m-k-1)\, u(k) - S + 1 \leq 0$$

and we have an initial feasible point. In order to have an optimal feasible solution to the original problem, the value of S must be 0 at the constrained optimum. This can be done by modifying the cost functional J by adding the penalty term $500S^2$. The cost functional thus becomes

$$J = \Delta t \sum_{i=0}^{m-1} u(i)^2 + 500S^2$$

5.1.2 Solution Using the SUMT Method

The above method for obtaining an initial feasible point was adopted except that, since logarithmic barrier functions were used, the two constraints incorporating the slack variable were modified to prevent negative arguments:

$$-\Delta t^2 \sum_{K=0}^{m-2} (m-k-1)\, u(k) - S^2 + 1 \leq 0$$

$$-\Delta t \sum_{k=0}^{m-1} u(k) - S^2 \leq 0$$

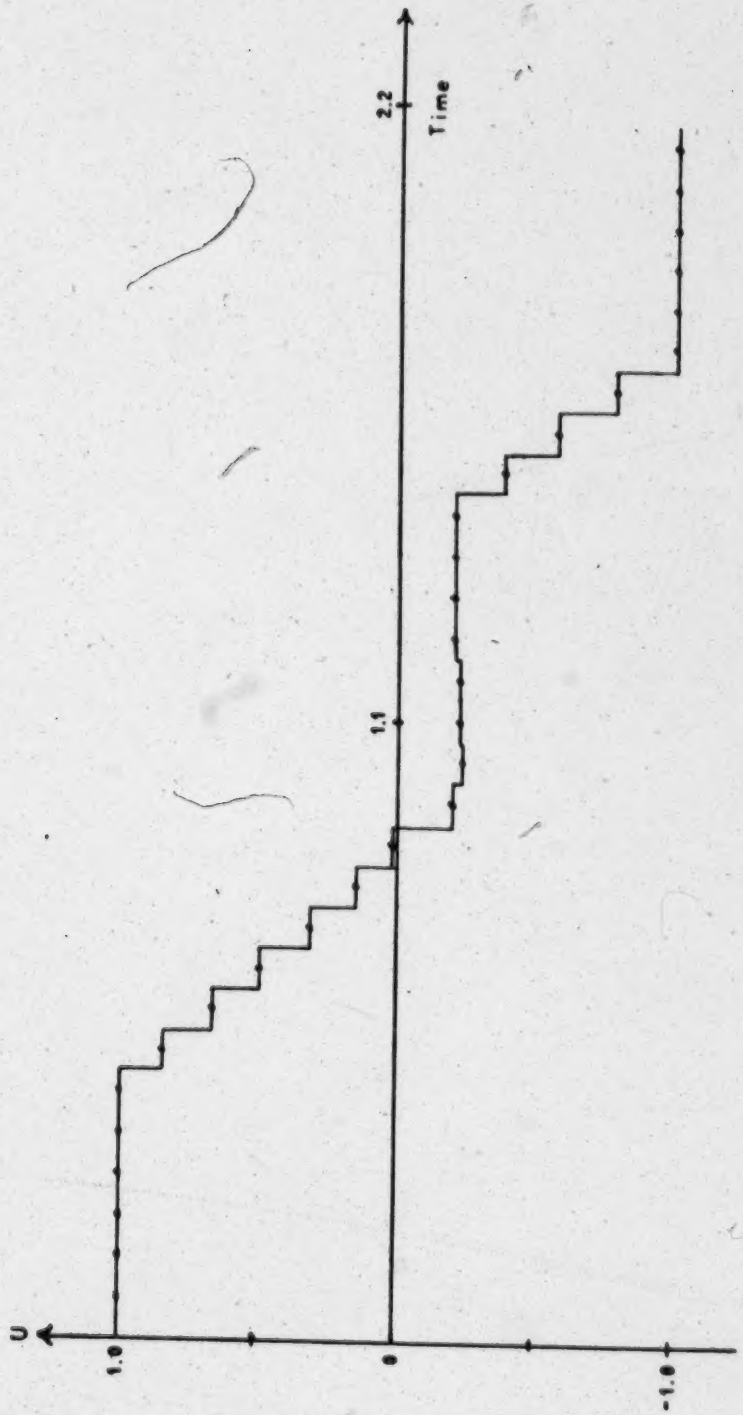

Figure 5.1 MER optimal control

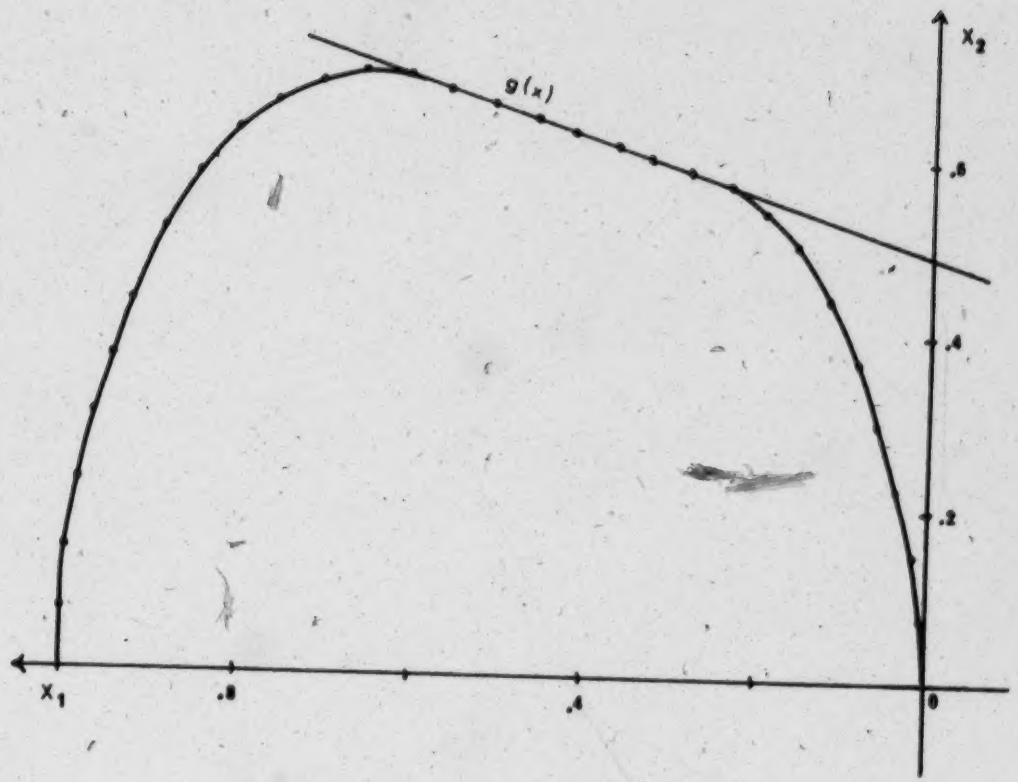

Figure 5.2 MER optimal trajectory

5.2 Numerical Results

The results of applying Rosen's GP and SUMT to the solution of the MER problem are presented in Table 5.1. The optimal trajectory and control are given in Figures 5.1 and 5.2.

TEST	PROB	SD	CPTAN	CG	DFP
GRAD	IT	30	25	24	24
PROJ	#F	39	29	27	27
	TIME	16.589	14.822	14.604	20.231
	IT	#	138*	>280*	159
SUMT	#F		848	>1296	634
	TIME		58.808	>120.0	58.759

m=30 time steps, r=1.0, c=5.0
\# - NO SOLUTION ATTEMPTED
* - INACCURATE ANSWER

Table 5.1 Numerical results: MER

Comments

All four implementations of GP (SD, PARTAN, CG, and DFP) solved the problem quickly and accurately for m=30 time intervals. Conversely the SUMT algorithm had a great deal more trouble, with CG being unable to produce a solution in a reasonable amount of execution time and the solution provided by PARTAN being highly inaccurate. Only DFP provided a reasonably accurate SUMT solution in a practical length of time.

As expected for the GP solution, SD had the largest number of iterations and DFP possessed the largest execution time, the objective function being a well-behaved quadratic. In general, the behavior of the accelerators was as detailed in 3.9.4 except that DFP had the smallest number of iterations. Again the advantage of using a direct method as compared to an indirect method was apparent.

The solution provided by the DFP SUMT algorithm required 634 function evaluations and 3 times the amount of execution time.

Additionally it was much easier to calculate analytically the function and and gradient required for the GP algorithm than it was for the SUMT version (another common advantage of this method of solution).

Conclusion

These results suggest that this implementation of GP incorporating several accelerators is an effective tool for the mathematical programming solution of control problems with linear constraints and in general is superior for this type of problem to simple implementations of penalty-barrier algorithms.

5.3 Earth-Mars Orbit Transfer Control Problem

The second sample control problem is a non-linear system with a minimum time cost functional, which will be solved using the epsilon technique [39]. This problem is the normalized Earth-Mars orbit transfer solved previously in a paper by Moyer and Pinkham [28] and by Taylor and Constantinides using the epsilon technique. The orbit transfer is to be performed using minimum fuel, but for a constant-thrust ion jet this is equivalent to minimum time. The control variable is the thrust angle Θ. The system equations are:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = x_3^2 - (\mu / x_1^2) + (T \sin\Theta) / (m_1 + \dot{m} t)$$

$$\dot{x}_3 = -(x_2 x_3 / x_1) + (T \cos\Theta) / (m_1 + \dot{m} t)$$

The boundary conditions and constants are

$$x(t_0) = \begin{bmatrix} 1.0 \\ 0.0 \\ 1.0 \end{bmatrix}, \quad x(t_f) = \begin{bmatrix} 1.525 \\ 0.0 \\ 0.8098 \end{bmatrix}$$

$$\begin{bmatrix} \mathcal{H} \\ m_1 \\ T \\ \dot{m} \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \\ 0.1405 \\ -0.0749 \end{bmatrix}$$

The epsilon cost functional is

$$J(\Theta,\epsilon) = (1/2\epsilon)\int_{t_0}^{t_f} \| x - f(x,\Theta,t) \|^2 dt + \int_{t_0}^{t_f} dt$$

The geometry of the problem is given in Figure 5.3.

The state variables and control variable will be expanded by the Rayleigh-Ritz (RR) method. The easiest and most obvious function to use in the expansion is the function $\sin(i\pi t/t_f)$. The discrete representation is now:

$$x_{in} = x_{i1} + (x_{if} - x_{i1})(n-1)/(N-1) + \sum_{j=1}^{M} a_{ij} \sin(\omega_j)$$

$$\dot{x}_{in} = (x_{if} - x_{i1})/(N-1)\Delta t + \sum_{j=1}^{M} \frac{j\pi}{(N-1)\Delta t} a_{ij} \cos(\omega_j)$$

$$\Theta_n = \Theta_1 + (\Theta_n - \Theta_1)(n-1)/(N-1) + \sum_{j=1}^{M} b_j \sin(\omega_j)$$

where

N=total number of time points

n=time point under consideration

M=number of terms in the Rayleigh-Ritz expansion

$\omega_j = [\, j\pi(n-1)/(N-1)\,]$

$t_f = (N-1)\Delta t$

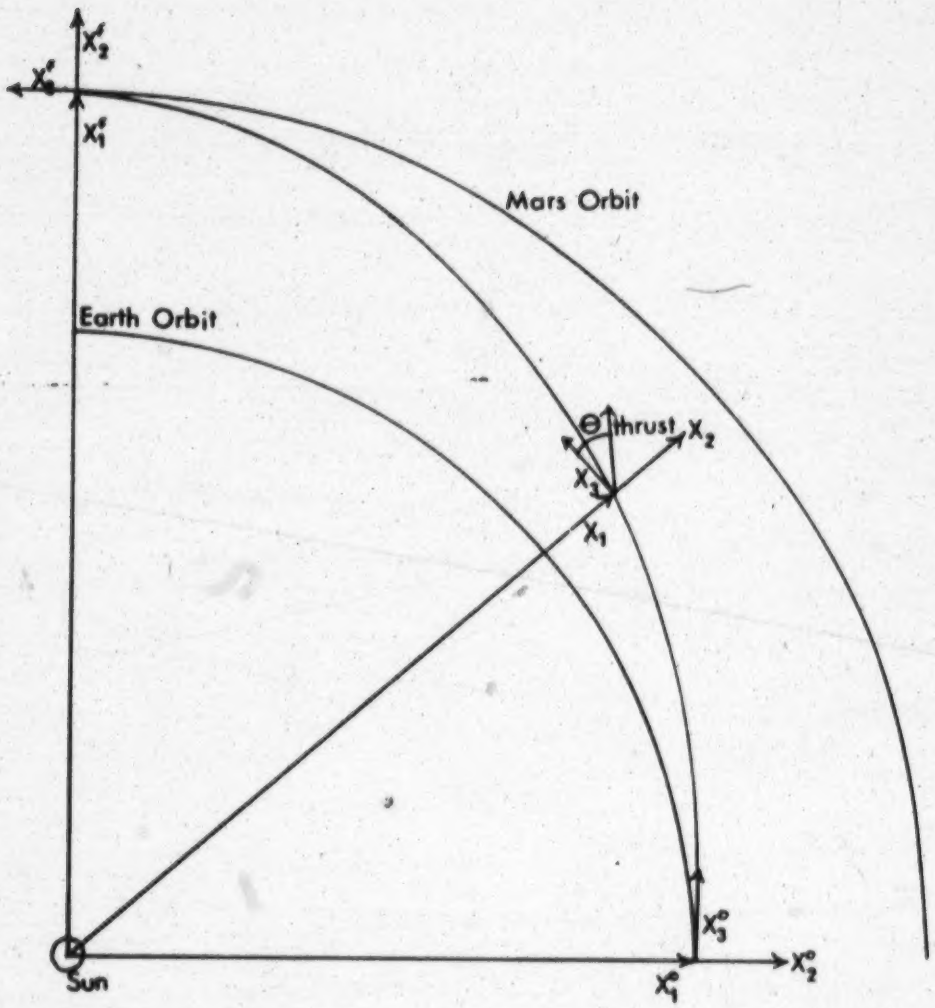

Figure 5.3 Geometry of Earth-Mars orbit transfer

The parameters a_{ij}, b_j, θ_i, θ_f, and Δt are the variables to be solved for.

Let the epsilon cost functional be expressed in the form

$$J(\theta,\epsilon) = \delta^T \delta$$

where δ is a (3N+1) column vector given by

$$\delta(z) = [\{\sqrt{\Delta t/\epsilon}\,\delta\dot{x}_{1n}\}_{n=1}^{N}, \{\sqrt{\Delta t/\epsilon}\,\delta\dot{x}_{2n}\}_{n=1}^{N}, \{\sqrt{\Delta t/\epsilon}\,\delta\dot{x}_{3n}\}_{n=1}^{N}, ((N-1)\Delta t)^{M}]$$

and

$$\delta\dot{x}_{1n} = \dot{x}_{1n} - x_{2n}$$

$$\delta\dot{x}_{2n} = \dot{x}_{2n} - (x_{3n}^2/x_{1n}) + (1/x_{1n}^2) - (T\sin\theta_n)/(m_1 + \dot{m}(n-1)\Delta t)$$

$$\delta\dot{x}_{3n} = \dot{x}_{3n} + (x_{2n}x_{3n})/x_{1n} - (I\cos\theta_n)/(m_1 + \dot{m}(n-1)\Delta t)$$

and z is the (4M+3) column vector of variables

$$z = [a_{11}, \ldots, a_{1M}, a_{21}, \ldots, a_{2M}, a_{31}, \ldots, a_{3M}, b_1, \ldots, b_M, \theta_i, \theta_f, \Delta t]$$

(The above formulation, with corrections, is from the paper by Taylor and Constantinides [39])

The resulting problem can be solved in three different ways:

(a) The Gradient Technique: This is a standard method for solving control problems. However Taylor and others [3,41,42] have been unable to obtain satisfactory results using this method. A solution by this method was not attempted.

(b) The Modified Newton-Raphson Method (MNR): This method is mentioned by Balakrishnan [3] and developed by Taylor et al [41] and was used by Taylor and Constantinides [39] to solve this problem. A development of it will be given below.

(c) Other Mathematical Programming Techniques: This problem can be solved using algorithms such as DFP, conjugate gradients etc. A short development of these solution techniques will be given below.

5.3.1 Solution by the Modified Newton-Raphson Method (MNR) [40]

Using this method the cost functional $J(\theta,\epsilon)$ is expressed as $\delta^T\delta$. The vector δ is linearized in terms of the change of the problem variables, Δz. Thus

$$\delta = \delta_0 + F_z(\delta)\Delta z$$

$$J = \delta_0^T\delta + 2\delta_0^T F_z \Delta z + \Delta z^T F_z^T F_z \Delta z$$

where z was defined earlier.

$F_z(\delta)$ is the Jacobian matrix of δ. The minimization is accomplished by iteratively solving

$$z_{i+1} = z_i + \Delta z_i$$

Setting the gradient of $J(\theta,\epsilon)$ to zero

$$J_{\Delta z} = 0 = 2\delta_0^T F_z + 2\Delta z_i^T F_z^T F_z$$

and solving for the difference Δz_i one gets

$$\Delta z_i = -[F_z(\delta_i)^T F_z(\delta_i)]^{-1} F_z(\delta_i)^T \delta_{i-1}$$

5.3.2 Solution by the MNR Method and Golden Search

A small variation of the MNR was attempted whereby the stepsize was optimized for the minimum value of J. This optimization was done using the Golden Search.

5.3.3 Solution by Other Mathematical Programming Techniques

Again the cost functicnal $J(\Theta,\epsilon)$ is expressed as $\delta^T\delta$. One can use first order gradient techniques to minimize $J(\Theta,\epsilon)$ after calculating the gradient as

$$J(\Theta,\epsilon) = \delta^T\delta$$

$$\frac{\partial J}{\partial \epsilon}(\Theta,\epsilon) = \delta^T\frac{\partial \delta}{\partial \epsilon} + \left(\frac{\partial \delta^T}{\partial \epsilon}\right)\delta$$

$$= 2\,\delta^T\left(\frac{\partial \delta}{\partial \epsilon}\right)$$

The problem is now in a form readily solved by such techniques as DFP, conjugate gradients, etc.

5.3.4 A Variation tc the Solution by the Modified Newton Raphson Method (VMNR)

There is a second method due to Taylor and Constantinides [40] cf representing the problem. This method, with fewer variables, consists of expanding only the state variables by the Rayleigh-Ritz method and then performing an ordinary minimization for the control variables at each discrete point along the state trajectory. The problem thus reduces to

$$z=[a_{11}, a_{12}, \ldots, a_{1M}, a_{21}, \ldots, a_{2M}, a_{31}, \ldots, a_{3M}, \Delta t]$$

with the state variables represented as before. The optimal control is then obtained at each point along the discrete trajectory by minimizing the dynamic error with respect to the control variables. Thus setting

$$\frac{\partial}{\partial \theta}(\|\dot{x}-f(x,\theta,t)\|^2)=0$$

yielding $\theta=\tan^{-1}\{\dot{x}_2-(x_3^2/x_1)+(\mu/x_1^2)\}/\{\dot{x}_3+(x_2x_3/x_1)\}$ at each point along the trajectory.

5.4 Numerical Results

This problem was solved using the four techniques discussed above. The results of applying these techniques to the solution of the Earth-Mars orbit transfer problem are presented in Table 5.2. The optimal trajectories and controls produced by each method are given in Figures 5.4 to 5.11.

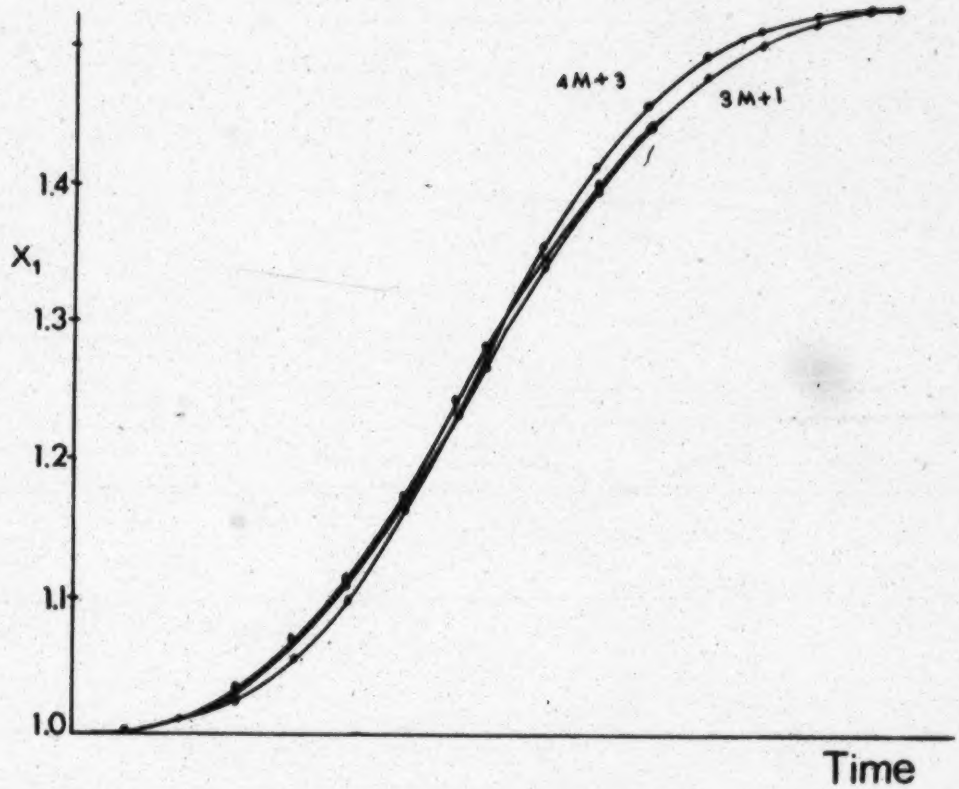

Figure 5.4 State trajectory x_1

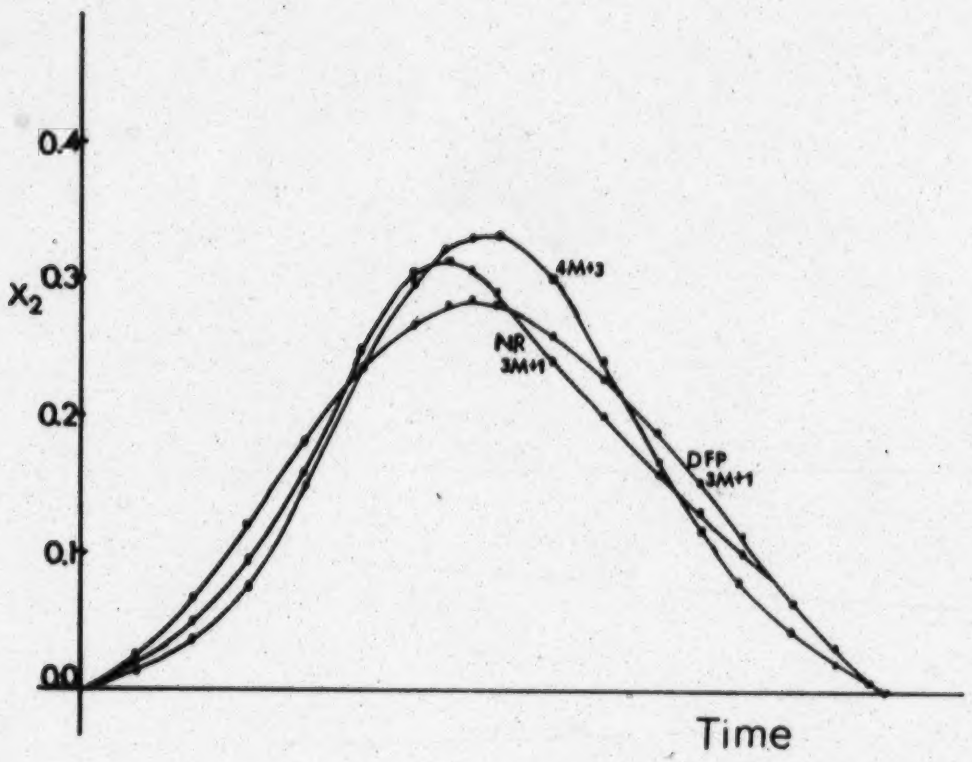

Figure 5.5 State trajectory x_2

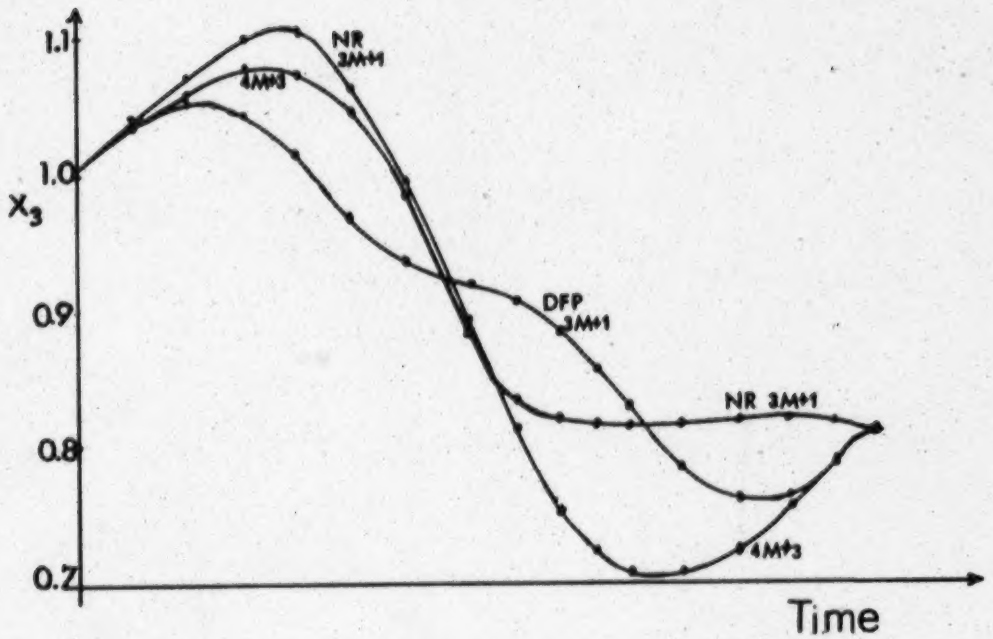

Figure 5.6 State trajectory x_3

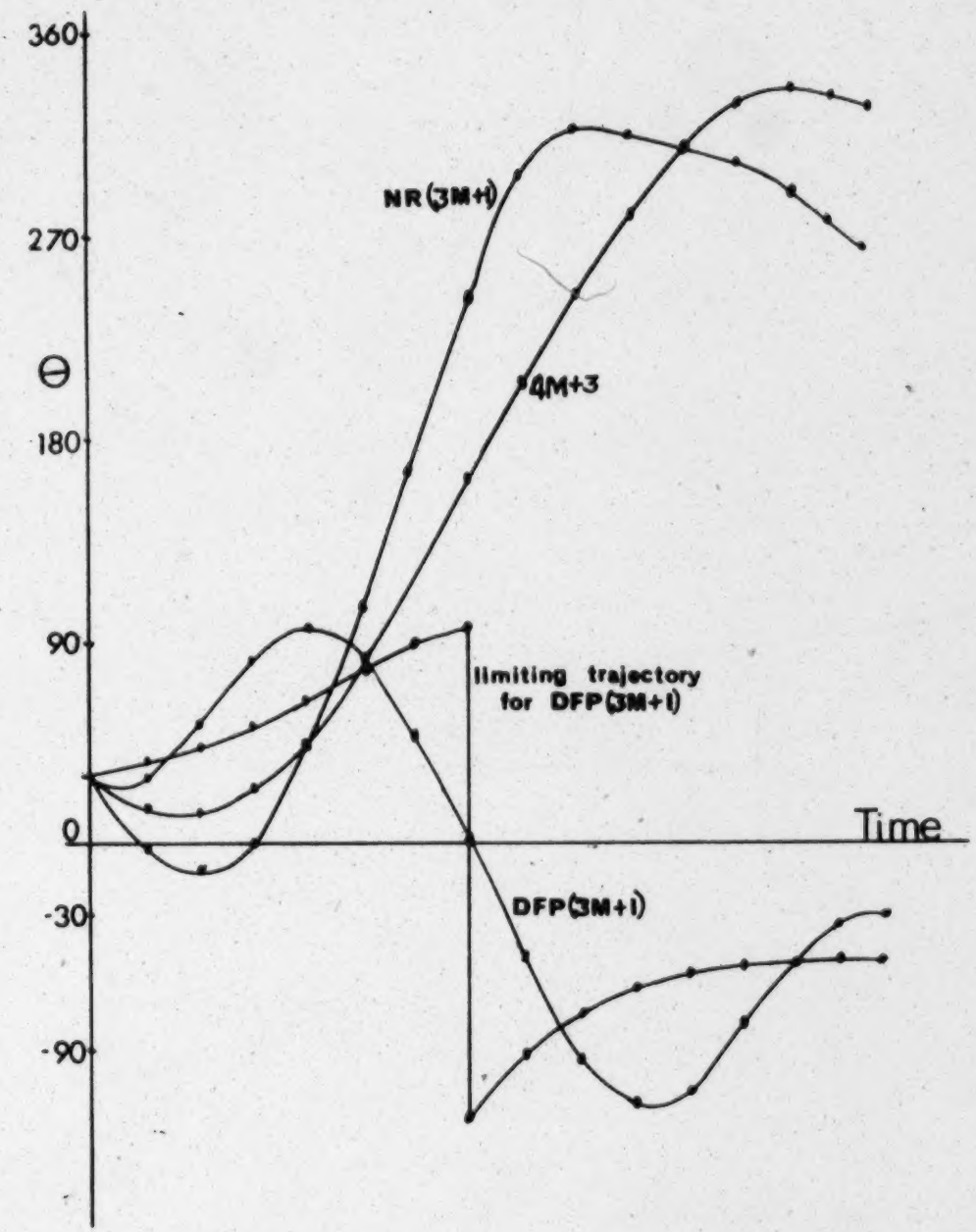

Figure 5.7 Optimal control angle

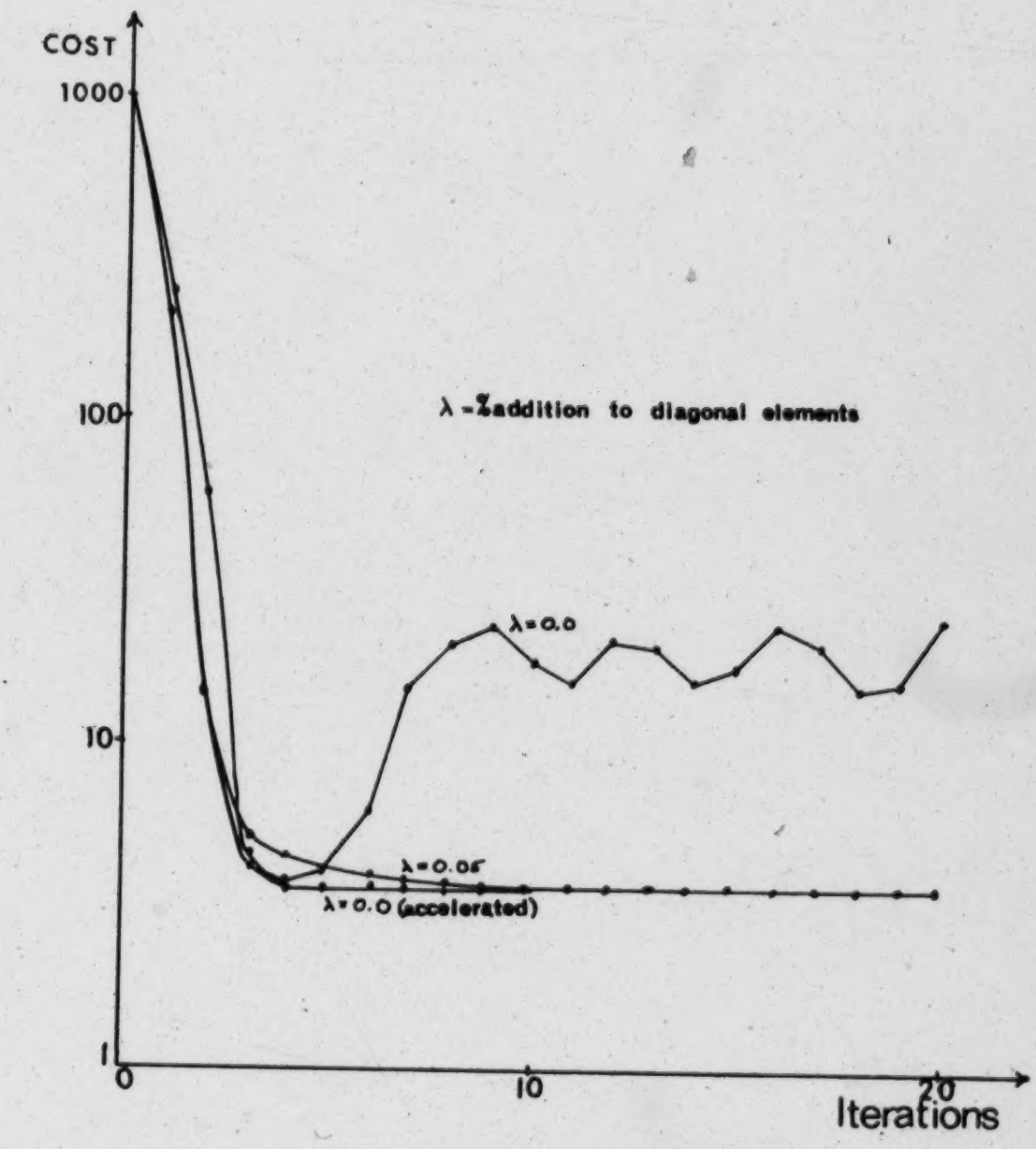

Figure 5.8 Cost versus iterations - NR minimization

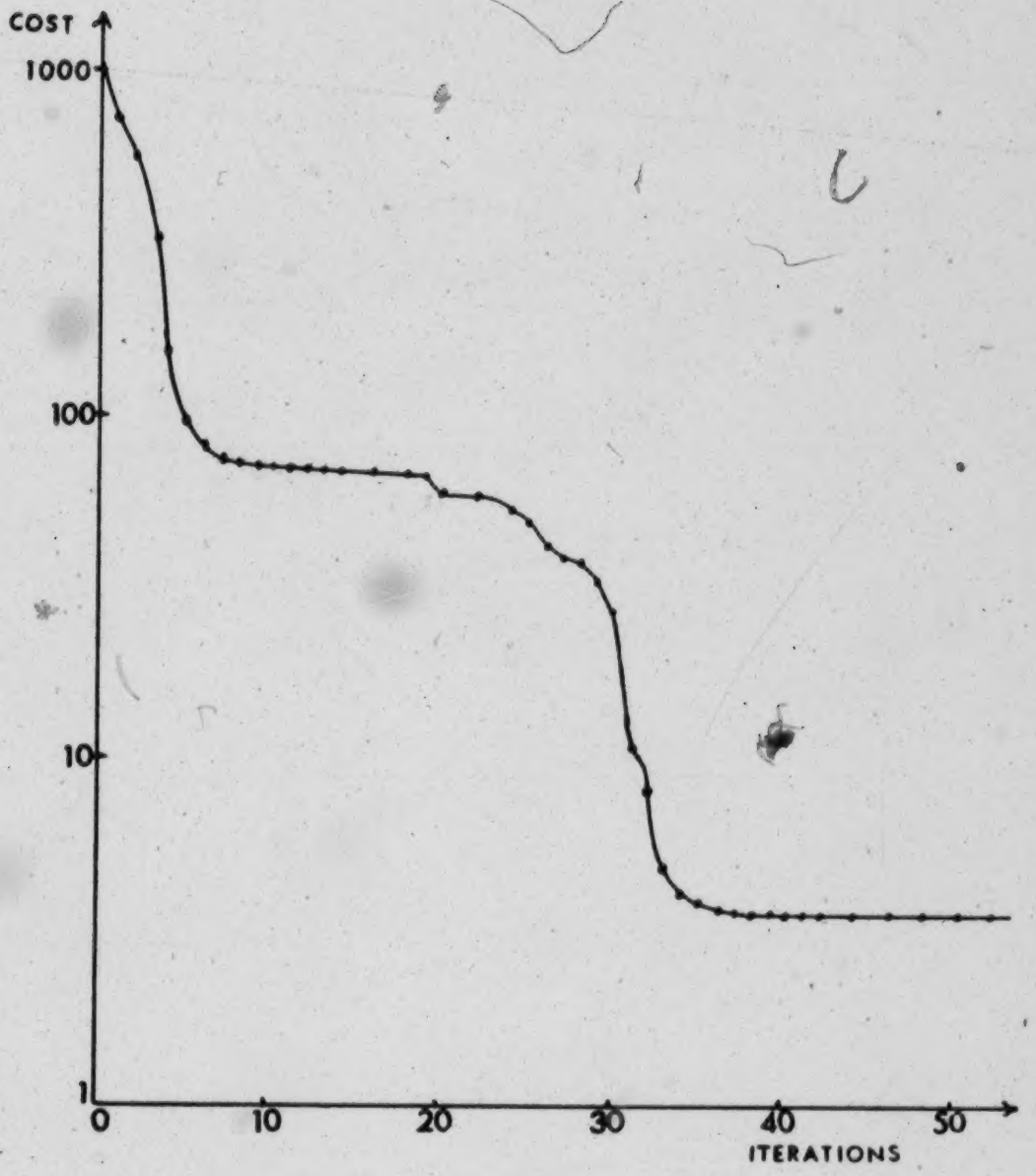

Figure 5.9 Cost versus iterations - DFP minimization

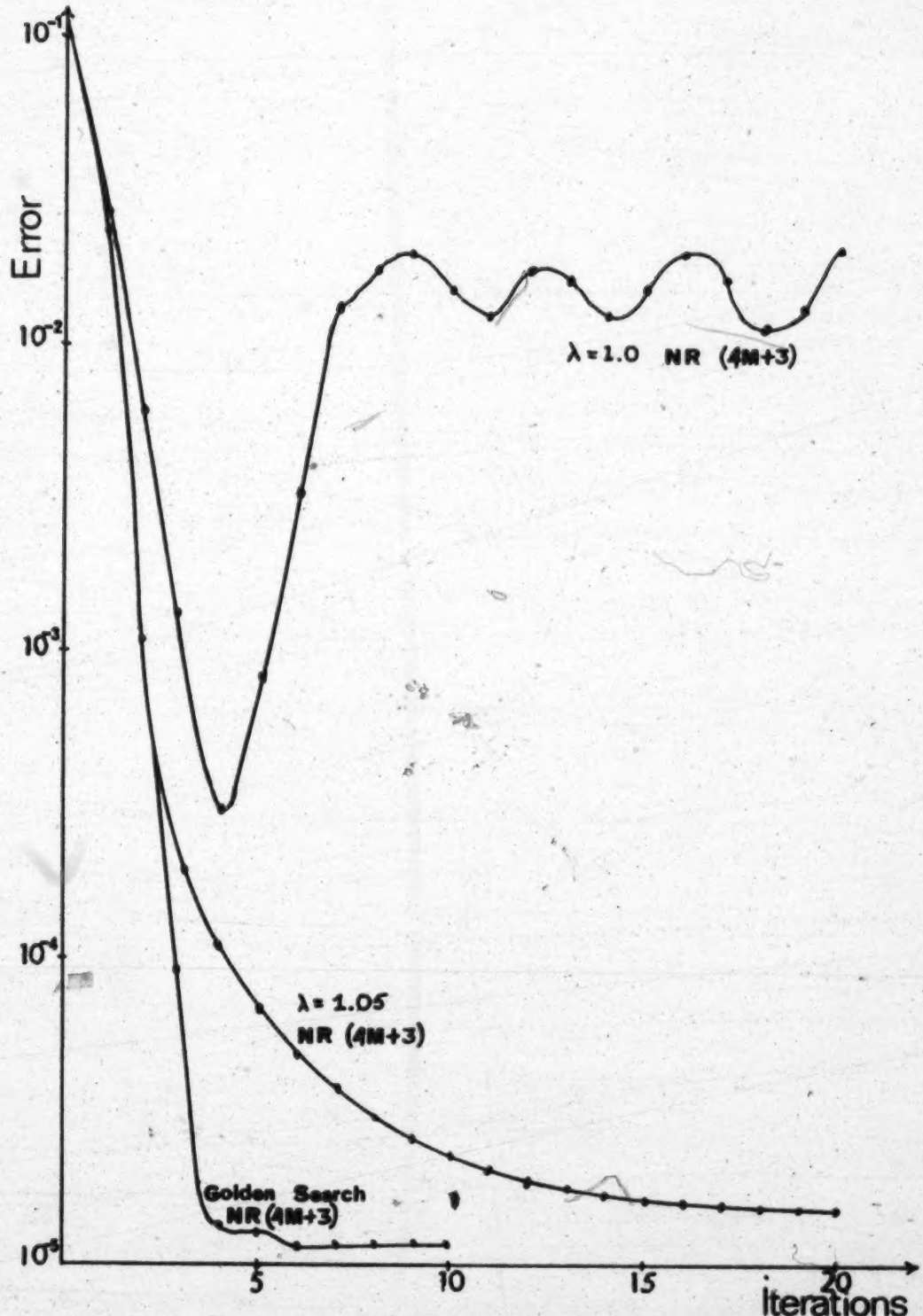

Figure 5.10 Error versus iterations - NR minimization

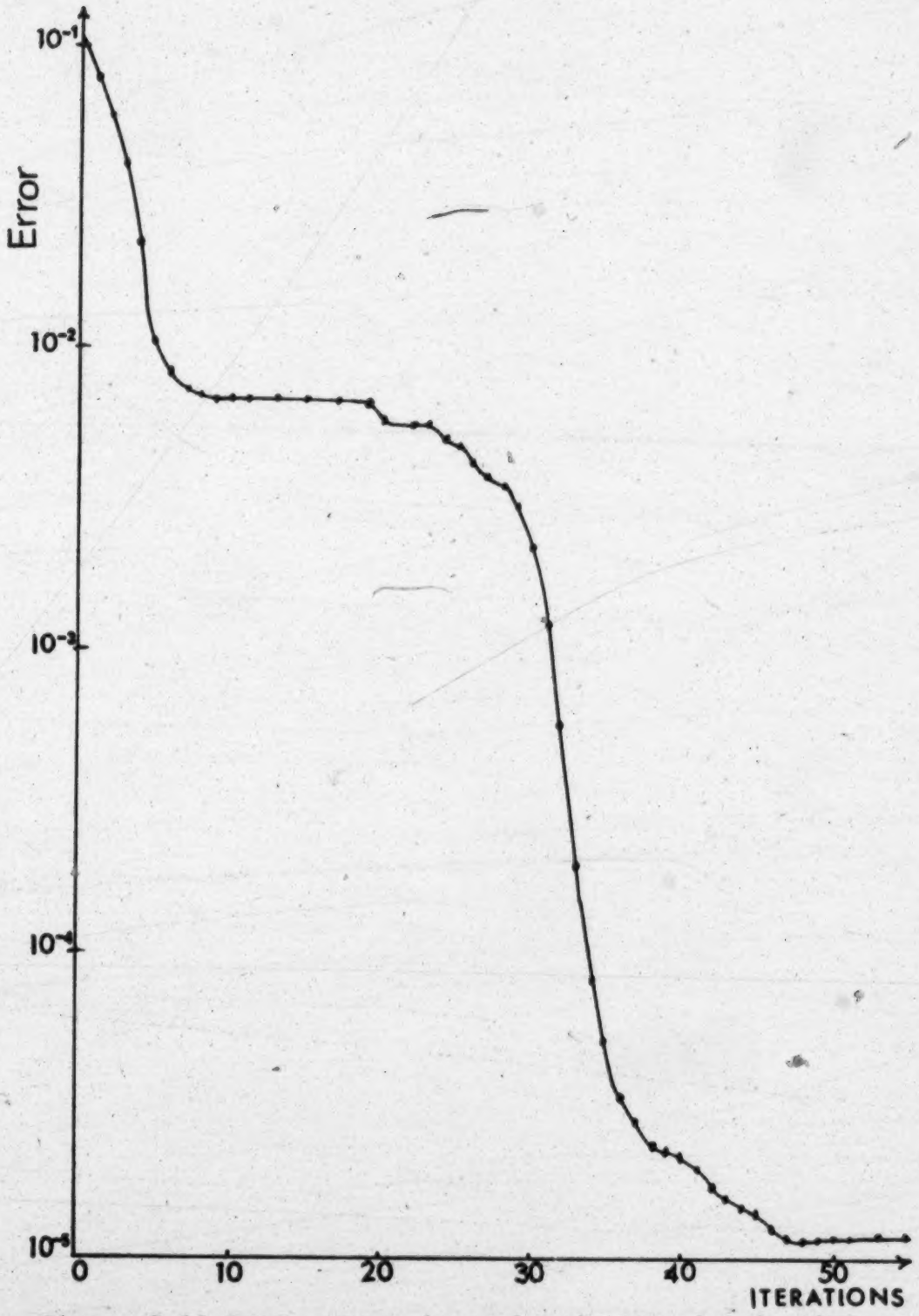

Figure 5.11 Error versus iterations - DFP minimization

Comments

Taylor and Constantinides' results for N=30, m=7, and $\epsilon = 10^{-4}$ were reproduced for the MNR method. However in order to achieve similar results, the technique (mentioned by Taylor et al. [42]) of increasing the diagonal elements of the matrix $[F_2^T F_2]$ by a small percentage had to be adopted. Without this modification the solutions did not converge and oscillated wildly due to the near singularity of this matrix. The value of the increment was not critical in this case and 5% was chosen. The number of iterations was set at 20 and the initial control (θ_0, θ_1) was set to $(0, 2\pi)$ since this gave reasonable results for the DFP solutions and computing cost was a factor which inhibited wide experimentation. The DFP algorithm was used to provide all other sclutions presented. The other algorithms performed worse, with CG taking twice the execution time to provide a solution and with PARTAN taking even longer and not converging to an acceptable solution. SSVM was comparable, especially if restarted.

Relative Computing Costs

For the formulation with the control variable expanded as a FR series the MNR and DFP solutions were almost identical. However the DFP solution took less than half the execution time and 50% less memory. This is due to the $(4M+3)\times(4m+3)/2$ matrix inversion required for the NR

solution. This situation becomes even greater in the DFP solutions' favour as the number of variables is increased because most of the cost of the MNR method is in this inversion. This is expensive even though advantage is taken of the fact that the matrix is symmetric positive-definite.

The method employing the Golden Search as a single variable search worked and produced solutions almost identical (within .01%) to the DFP solutions for the (4m+3) variable problem. The number of iterations was half that of the MNR method and required only approximately 60% of the execution time. This accelerator technique did not work if the elements of the diagonal were modified as in the standard MNR method, the solution times being the same in both cases. The tolerance of convergence chosen for the Golden Search was a crude 0.4.

The Golden Search did not work for the VMNR method, causing the problem to converge to a non-optimal solution. The technique of incrementing the diagonal elements also was unsuccessful for this method, again causing convergence to the same non-optimal solution. However the "convergence" was more rapid.

The VMNR method of solution was not found to be very satisfactory. The control was very sensitive to changes in t_ϕ, m, and ϵ and varied significantly as these were changed. Also while the cost per iteration was less, a larger number

of iterations was required for a solution of comparable accuracy.

In addition, this formulation possessed at least two other local minima to which the method often converged. This fact plus the lack of information in the article [40] made it impossible to achieve comparable results. As a consequence some of the conclusions reached here differ from those given by Taylor and Constantinides for this method [40]. These new conclusions are:

(1) The convergence of the algorithm requires more iterations for a comparable control and accuracy, though the cost per iteration is less.

(2) The dynamic error is greater resulting in a less accurate control, though this error can be reduced by more iterations.

(3) This method created at least two other non-optimal solutions to which the method had a tendency to converge.

The DFP solution of the formulation involving (3m+1) variables was even more sensitive, the solution time for a comparable cost being approximately the same as for the VMNR method. The VMNR solutions for this problem tended toward the control trajectory shown in Figure 5.7. This trajectory is similar to that produced by Moyer and Pinkam [28] for their solutions to the Earth-Mars orbit transfer problem produced by the Gradient and Second Variation techniques.

The main disadvantage of the DFP solution of these problems was that the solution time was sensitive to the initial boundary conditions. The solutions times of the MNR method were relatively unaffected by changes in ϵ, initial total time, and initial values of the control; on the other hand, the DFP solution time was affected by changes in these values and solutions both much better and much worse than those given were achieved. The initial conditions chosen represented the mean values and also were a logical initial guess.

The choice of the parameter ϵ had a great effect on the rate of convergence of the DFP solutions. For large ϵ the DFP algorithm converged much more rapidly than for a smaller value of ϵ. However, as in the NR case, the error in the solution is larger for large ϵ.

CHAPTER VI

6. Summary and Conclusion

This study has been concerned with the review, implementation and evaluation of standard nonlinear mathematical programming techniques and their application to the solution of certain mathematical programming and optimal control problems.

6.1 Summary of Results

In Chapter 2 the methods used in solving unconstrained problems were reviewed and implemented practically. These computer implementations were applied to the solution of a family of test problems. The solutions were then compared and the relative merits of the algorithms were discussed. The conclusions drawn agreed generally with those of other authors in this field except that the results were often found to be highly dependent on the mode of implementation and to a lesser extent dependent on the problem under consideration. From this it can be inferred that there is no way of determining a priori which method will be best for solving a particular problem, but only that some methods are more likely to prove more successful than others.

The solution of constrained problems was the topic of study of Chapter 3. In this Chapter the solution by penalty and barrier methods, including SUMT, was discussed and then applied to the solution of a set of test problems. The

solutions of these problems by the different methods using the minimization techniques of Chapter 2 were then compared. This comparison revealed the defects inherent in these methods and the weaknesses of first order gradient minimization techniques when applied to the solution of penalty constrained problems by these techniques.

Rosen's gradient projection algorithm was described along with an implementation of accelerating techniques which improved its rate of convergence. These accelerators consisted of the PARTAN, conjugate gradient and DFP algorithms. The accelerators, particularly CG and PARTAN, were found to improve substantially the rate of convergence in comparison with the original steepest descents direction generation.

The performance of gradient projection was also compared to that of SUMT and it was shown that for a problem with linear constraints GP is more efficient.

Chapter 4 reviewed two methods of transforming the optimal control problem into a mathematical programming problem. The first method consisted of discretizing the control problem and approximating the state equations by difference equations. The second method discussed was Balakrishnan's epsilon technique, a method of optimization which avoids the explicit solution of the dynamic equations.

Chapter 5 consisted of applying the mathematical

programming techniques of Chapters 2 and 3 and the methods of Chapter 4 to the solution of two selected control problems. These problems were the minimum energy regulator problem and the Earth-Mars orbit transfer problem. MER was solved by means of Rosen's gradient projection and SUMT with the two methods being compared. The Earth-Mars orbit transfer problem was solved by means of a modified Newton-Raphson technique and by the DFP algorithm. The two methods of solution were compared with the superiority of the DFP algorithm for this type of problem (possessing a large number of variables) being demonstrated.

6.2 Conclusions

In the course of this study, possible areas of further investigation became apparent. Relatively little work has been done in applying mathematical programming to the design of optimal control systems. More particularly, mathematical programming techniques could be used in applying Balakrishnan's epsilon technique to a wider range of control problems, especially those whose cost function is something other than minimum time. More work could also be done on extending the types of Ritz expansions used in the epsilon technique, especially to the use of expansions other than the sine series.

Another area of interest is on-line use of the epsilon technique in the generation of controls for dynamic

processes. This would entail studying the accuracy of the controls generated, the speed of solution and the minimum memory requirements necessary for acceptable performance.

BIBLIOGRAPHY

[1] Aoki, M., "Introduction to Optimization Techniques," MacMillan, 1971.

[2] Balakrishnan, A.V., "On a new computing technique in optimal control," SIAM J. Contr.,6,pp.149 -173, May 1968.

[3] ------, "On a new computing technique in optimal control and its application to minimal time flight profile optimization," J. Optimiz. Theory Appl., 4, pp.1-21, 1969.

[4] Bellman, R., "Dynamic Programming," Princeton University Press, 1957.

[5] Bryson A.E., and Ho, Y.C., "Applied Optimal Control," Blaisdell, 1969.

[6] Courant, R., "Variational methods for the solution of problems of equilibrium and vibrations," Bull. Am. Math. Soc., 49, pp. 1-23, 1943.

[7] Davidon, W.C., "Variable Metric Method of Minimization," Report ANL-5990(rev.), Argonne National Laboratory, 1959.

[8] Denham, W.F.,Bryson, A.E., "Optimal programming problems with inequality constraints II: solution by steepest - ascent," AIAA Jour., 2, pp. 25-33, 1964.

[9] Fiacco, A.V., and McCormick, G.P., "The sequential unconstrained minimization technique for nonlinear programming, a primal-dual method," Management

Science, 10, pp. 360-366, Jan., 1964.

[10] ------, "Computational algorithm for the sequential unconstrained minimization technique for nonlinear programming," Management Science, 10, pp. 601-617, July 1964.

[11] ------, "Extension of SUMT for nonlinear programming: equality constraints and extrapolation," Management Science, 12, pp. 816-828, July, 1966.

[12] ------, "Nonlinear Programming: Sequential Unconstrained Minimization Techniques," Wiley, 1968.

[13] Fletcher, R., and Powell, M.J.D., "A rapidly convergent descent method for minimization," Computer J., 6, pp. 163-168, 1963.

[14] Fletcher, R., and Reeves, C.M., "Function minimization by conjugate gradients," Computer J., 7, pp. 149-154, 1964.

[15] Fox, R.L., "Optimization Methods for Engineering Design," Addison-Wesley, 1971.

[16] Goldfarb, D., 'Extension of Davidon"s variable metric method to maximization under linear inequality and equality constraints," SIAM Jour. of Applied Math., 17, pp. 739-764, 1969.

[17] Gelfand, I.M., Fomin, S.V., "Calculus of Variations," Prentice-Hall, 1963.

[18] Hauer, J.F., "A TPBVP solution for minimum energy control of the system $\dot{x}=(x, u)$," (mimeographed notes), Department of Computing Science, University

of Alberta, 1970.

[19] -----, "The MER with state constraints," (mimeographed notes), Department of Computing Science, University of Alberta, 1970.

[20] ------, "Parameter Optimization Theory," Technical Report TR72-1, Department of Computing Science, University of Alberta, 1972.

[21] Huang, H.Y., and Levy, A.V., "Numerical experiments on quadratically convergent algorithms for function minimization," J. Optimiz. Theory Appl., 6 pp. 269-282, 1970.

[22] Jacoby, S.L.S., Kowalik, J.S., and Pizzo, J. T., "Iterative Methods for Nonlinear Optimization Problems," Prentice-Hall, 1972.

[23] Kirk, D.E., "Optimal Control by Mathematical Programming," Prentice-Hall, 1971.

[24] Kunzi, M.P., Krelle, W., and Cettli, W., "Nonlinear Programming," Blaisdell, 1966.

[25] Lasdon, L.S., Mitter, S.K., and Warren, A.D., "The conjugate gradient method for optimizal control problems," IEEE Trans. Automatic Control, AC-12, pp. 132-138, April 1967.

[26] Lasdon, L.S., Warren, A.D., Rice, R.K., "An interior penalty method for inequality constrained optimal control problems," IEEE Trans. on Automatic Control, AC-12, pp. 388-395, 1967.

[27] Lasdon, L.S., "Conjugate direction methods for optimal

control," IEEE Trans. on Automatic Control, AC-15, pp. 267-268, 1970.

[28] Moyer, H.G., and Pinkham, G., "Several trajectory optimization techniques-part 2: applications," in "Computing Methods in Optimization Problems," Balakrishnan and Neustadt, Eds., New York: Academic, pp. 91-105, 1964.

[29] Oren, S.S., and Luenberger, D.G., "The self-scaling variable metric algorithm (SSVM)," 5th. Hawaii International Conference on System Sciences, pp. 130-132, 1972.

[30] Pierre, D.A., "Optimization Theory with Applications," Wiley, 1969.

[31] Pierson, B.L., Rajtora, S.G., "Computational experience with the Davidon method applied to optimal control problems," IEEE Trans. on Systems Science and Cybernetics, SSC-6, pp. 240-242, 1970.

[32] Pontryagin, L.S., Boltyanski, V.G., Gamkrelidze, R.V., Mishkchenko, E.F., "The Mathematical Theory of Optimal Processes," Interscience, 1962.

[33] Rosen, J.B., "The gradient projection method for nonlinear programming, part I, linear constraints," J. SIAM, 8, pp. 181-217, 1960.

[34] Sage, A.P., "Optimum Systems Control," Prentice- Hall, 1968.

[35] Shah, B.V., Buehler, R.J., and Kempthorne, O., "Some algorithms for minimizing a function of several

variables," J. SIAM, 12, pp. 74-91, 1964.

[36] Sinotti Jr., J.F., Luenberger, D.G., "Solution of optimal control problems by the method of conjugate gradients," Prcc. 1967 JACC, pp. 566-574, 1967.

[37] Tabak, D., "Comparative study of various minimization techniques used in mathematical programming," IEEE Trans. on Automatic Control, AC-14, P. 572, 1969.

[38] Tabak, D., and Kuo, B.C., "Optimal Control by Mathematical Programming," Prentice- Hall, 1971.

[39] Taylor, J.M., and Constantinides, C.T., "Optimization: application of the epsilon method," IEEE Trans. on Automatic Control, AC-17, pp. 128-131, Feb., 1972.

[40] ------, "A computation aspect of the epsilon method," IEEE Trans. on Automatic Control, AC-17, P. 738, Oct., 1972.

[41] Taylor Jr., L.J., Smith, J.H., and Wiff, K.W., "A comparison of minimum time profiles for the F-104 using Balakrishnan's epsilon technique and the energy method," Symposium on Optimization, Nice, pp. 327-335, June 29- July 05, 1969.

[42] ------, "Experience using Balakrishnan"s epsilon technique to compute optimum flight profiles," J. Aircraft, 7, pp. 182-187, 1970.

[43] Tripathi, S.S., Narendra, K.S., "Optimization using conjugate-gradient methods," IEEE Trans. on Automatic Control, AC-15, pp. 268-270, 1970.

[44] Zangwill, W.I., "Nonlinear Programming: A Unified

Approach," Prentice-Hall, 1969.

[45] Zoutendijk, G., "Method of Feasible Directions," Elsevier, 1960.

[146] Zoutendijk, G., "Nonlinear programming: a numerical survey", J.SIAM Control, 4, pp. 194-210, 1966.

[47] Carrol, C.W., "The created response surface technique for optimizing nonlinear restrained systems", Operations Res., 9, pp.167-184, 1961.

[48] Rosen, J.B., "Iterative solution of nonlinear optimal control problems", J.SIAM Control, 4, pp. 223-244, 1966.

[49] Hauer, J.F., and Descheneau, J.C., "MORPAK-I: A mathematical optimization routines package for unconstrained minimization", Technical Report TR73-9, Department of Computing Science, University of Alberta, July, 1973.

[50] ------, "MORPAK-II" (to be published).